curso de basic

VOLUME 1

```
120 PRINT "NOME DO ARQU
130 PRINT FNPC$(11,7);
140 PRINT "
150 PRINT FNPC$(12,7);
160 PRINT " ";STRING$(7
170 LI=12:TC=8:GOSUB 30
180 IF AR$="" THEN AR$=
190 FOR F=1 TO LEN(AR$)
200    MR$=MID$(AR$,F,1)
210 NEXT F
220 AR$=AR$+".DAT"
230 '
240 ' DEFINE DEFAULT
250 '
260 M1$=" ":M2$=" ":M3
```

Piazzi - Carvalho Jr.

curso de basic

2ª EDIÇÃO

VOLUME 1

curso de basic

PIERLUIGI PIAZZI
LUIZ TARCÍSIO DE CARVALHO JR.

VOLUME 1

EXPEDIENTE

Coord. Editorial	Pierluigi Piazzi
Coord. Didática	Betty Fromer Piazzi
Editoração	L.Tarcísio de Carvalho Jr
Arte	Ana Lúcia Antico
Capa	Fernando Gastão Moretti
Produção	Rosa K. Fromer

ALEPH
Publicações e Ass. Pedagógica Ltda.
C.P. 20.707 CEP 01498
São Paulo SP
TEL 843-3202

Dados de Catalogação na Publicação (CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

Carvalho Júnior, Luiz Tarcísio de, 1955-
C329c Curso de BASIC MSX / Luiz Tarcísio Carvalho Jr.,
v.1- Pierluigi Piazzi. -- São Paulo : Aleph, 1987-
(Série didática)

Conteúdo: v. 1. Elementar.

1. BASIC (Linguagem de programação para computadores) 2. MSX (Computadores) I. Piazzi, Pierluigi, 1943- II. Título. III. Série.

87-1634 CDD-001.6424

Índices para catálogo sistemático:

1. BASIC : Linguagem de programação : Computadores : Processamento de dados 001.6424
2. MSX BASIC : Linguagem de programação : Computadores : Processamento de dados 001.6424

SUMÁRIO

NOTA DO EDITOR

Conversar com um computador não é tarefa fácil. Tente imaginar um "médium" possuído por vários espíritos que falem línguas diferentes: a cada momento você deve ter presente com qual espírito está falando e em que língua. O computador está ora sob controle de seu sistema operacional, ora sob as ordens de um programa Basic, ora executando uma sub-rotina em linguagem de máquina.

Quando, há alguns anos, fui convidado (ou melhor dizendo, intimado por amigos, vizinhos e parentes) a dar um curso de computação para crianças, fiquei preocupadíssimo pensando na dificuldade que teria para explicar à meninada (de 8 a 10 anos) toda esta multiplicidade de personagens e linguagens.

Por via das dúvidas comecei pelo caminho mais seguro: coloquei cada criança na frente de um computadorzinho bem pequeno, mas nem por isso menos complexo, e deixei que eles brincassem um pouco.

Após umas rápidas explicações sobre como ligar a máquina e digitar as teclas, entreguei a cada garoto um exemplar de uma revista na qual eu havia publicado um programa em Basic que criava joguinhos de boliche.

Neste momento, começou uma série initerrupta de sustos! Depois de meia hora, estavam todos digitando com uma desenvoltura que, em cursos para adultos, só aparece lá pela terceira ou quarta aula. Não só isso: um deles alterou o programa de maneira a ter direito a um número maior de bolas de boliche durante o jogo! Outro mudou um comando que fazia a bola piscar durante a trajetória e transformou o jogo de boliche

num programa de desenhar foguetes!

Em resumo, um susto a cada dez minutos. Cada vez mais perplexo, comecei a lembrar um famoso romance de Arthur Clake, "O Fim da Infância", onde a população de crianças da Terra sofre uma mutação que a transforma numa "super-mente" tão avançada que faz os pais parecerem homens de Neanderthal!

Será que o acaso me havia colocado no meio de um grupo de "geninhos"? Ou talvez mutantes?

Minha perplexidade foi aumentando nas aulas seguintes, quando percebi a pouca utilidade das explicações que eu dava dos comandos Basic. Nem a metade prestava atenção e os que tentavam entender tinham dificuldades. Entretanto, depois de digitar um programa-exemplo colocado no quadro negro, eles deduziam em poucos minutos o que o comando fazia e, pior ainda, o utilizavam corretamente num contexto diferente.

Comecei a pesquisar em livros e revistas, mas tudo que encontrei foram técnicas criadas para adultos e adaptadas para crianças (com toques pseudo-pedagógicos e a inevitável citação de Piaget). Minha experiência didática, obtida através de mais de 50 mil alunos ao longo de 25 anos de carreira, não estava me servindo para nada. Como entender o que acontecia com os meus "monstrinhos"?

Um dos garotos era meu filho e, olhando-o com admiração, de repente lembrei a admiração com que meu pai me olhou certa vez há mais de 30 anos atrás. Neste momento se deu o "estalo" e consegui esclarecer o mistério: nesta época meus pais emigraram para o Brasil e a admiração de que falei foi provocada pela facilidade com que eu, aos 11 anos,consegui aprender o Português.

Era isso: a garotada não estava aprendendo computação, estava conversando numa outra língua. Comentando isso com minha mulher, que é professora de Francês, percebi que a técnica mais adequada para se ensinar computação é a mesma que devemos usar no ensino de uma lígua estrangeira. Ela me ensinou que a última coisa a ser ensinada a um menino é a gramática: ele é a própria gramática.

O grande erro no ensino da computação, tanto para crianças como para adultos, é justamente querer ensinar as regras operacionais do computador através de teoria, exposições de sintaxe, ou até (e esta é a barbaridade mais atroz) explicar o funcionamento da parte eletrônica do computador. Seria o mesmo que ensinar uma língua estrangeira começando pelo funcionamento do cérebro e explicar toda estrutura fonética e

gramatical da língua, quando na realidade o que se precisa é de conversação.

Quinze dias em Londres fizeram mais pelo meu Inglês que centenas e centenas de horas de aula durante todo meu curso secundário.

O fato do computador hoje ter um preço bem acessível (os menores não chegam a 4 salários mínimos), faz com que a "conversação" possa ser realizada em casa durante todo o tempo necessário. Este tempo às vezes é até exagerado, tanto é que um dos fenômenos sociais mais recentes é o aparecimento das chamadas "Viúvas da Computação", que disputam a madrugada de seus maridos com estas "diabólicas máquinas".

Essa experiência com as crianças, além de extremamente gratificante, me foi de enorme utilidade: joguei no lixo todos os textos de ensino de computação e montei uma equipe para escrever livros e apostilas que aproximam o usuário da máquina com as mesmas técnicas que se usam para aproximar alguém de uma nova língua. "O Fim da Infância" ainda não foi desta vez!

Pierluigi Piazzi

NOTA AOS PAIS E MESTRES

Este é um livro didático que pode ser utilizado tanto por auto-didatas que queiram se iniciar na linguagem BASIC, quanto por professores ou pais que queiram adotá-lo como livro texto em um "CURSO DE BASIC" para jovens e adultos.

Em ambos os casos, os autores partiram do pressuposto de que o aluno dispõe de um micro MSX, instalado e funcionando para poder aprender interagindo com o computador. Dispensa-se o uso de quaisquer periféricos a não ser um gravador cassete (para armazenar os programas digitados) e um terminal de vídeo que pode ser um simples televisor preto e branco: o uso das cores é discutido em apenas uma aula e o assunto pode ser evitado sem prejuízo de compreensão dos outros itens.

No caso de uma classe, é aconselhável que cada aula utilize aproximadamente 2 horas do tempo dos alunos, não colocando mais que dois deles à frente de

cada máquina. É interessante, também, que lhes seja reservado um tempo de máquina, além da aula propriamente dita, para que possam elaborar a resposta de seus exercícios sempre interagindo com o computador.

Como o material é auto-instrutivo, o papel do professor consistirá, em cada aula, numa rápida explanação prévia do assunto, seguido de um monitoramento das atividades de cada grupo, que desenvolverá as tarefas na velocidade de assimilação que lhe é própria.

Para os grupos mais velozes, o tempo de aula restante poderá ser usado para iniciar (ou até eventualmente completar) os exercícios propostos.

É importante salientar que neste volume não se pretende ensinar "programação", mas sim simplesmente "alfabetizar" os alunos no BASIC para que possam prosseguir seus estudos auxiliados por esse maravilhoso instrumento didático que é o microcomputador MSX.

No volume 2 daremos mais ênfase à "programação" propriamente dita e, em outros títulos da série, mostraremos como usar o micro para estudar línguas, matemática, ciências, desenho e, até, música.

Em toda série, porém, a preocupação fundamental é a de se utilizar o micro sem periféricos ou "softwares" especiais, bastando que o aluno tenha sido "alfabetizado" no BASIC para que tenha condições de digitar, com conhecimento de causa, curtos programas demonstrativos.

Se bem utilizado, um microcomputador pode abrir a porta de um universo maravilhoso, tanto para os alunos, quanto para os professores, pois permite a assimilação de matérias de estudo através de técnicas insuspeitadas até o aparecimento desta máquina.

Esperamos que este primeiro volume possa ser a chave desta porta mágica e revolucionária.

AULA 1

Assunto: Introdução ao Basic MSX.

Objetivo: Familiarizar o leitor com o teclado do computador.

INTRODUÇÃO AO BASIC MSX

Para poder controlar um computador você deve dar instruções numa linguagem que ele "entenda". O padrão MSX, assim como a maioria dos microcomputadores pessoais, entende várias linguagens como COBOL, ASSEMBLY, C, LISP, BASIC, etc... Para que ele entenda uma determinada linguagem ela deve ser "traduzida" para sua própria linguagem de máquina através de um programa específico.

Uma dessas linguagens, porém, já vem "carregada" de fábrica e o computador já a entende assim que é ligado: é o BASIC (Beginner's All Purpose Symbolic Instruction Code).

Essa linguagem apresenta vários dialetos que diferem entre si nas diferentes máquinas. O BASIC MSX, desenvolvido pela Microsoft, é constituído por um conjunto de palavras do idioma Inglês e determinadas regras de sintaxe com as quais você pode instruir o computador a realizar inúmeras funções.

Antes de começarmos nossas aulas de "conversação" em Basic, é conveniente nos familiarizarmos com a máquina em si.

Dessa forma, poderemos localizar facilmente as teclas que devemos digitar quando formos solicitados pelo texto. Aliás, todo este livro parte do pressuposto que o leitor está com um microcomputador MSX à sua frente, ligado e pronto para operar.

O TECLADO

Para nos comunicarmos com o computador deve haver um modo de lhe fornecermos instruções e outro

para que ele possa responder.

Geralmente, o teclado é o meio que utilizamos para dar instruções ao computador e a tela de TV é o que ele utiliza para respondê-las.

Quando você liga o microcomputador são apresentadas algumas mensagens de identificação do sistema e após alguns segundos deve aparecer a palavra "Ok". Esta é a indicação de que o computador está pronto para aceitar os seus comandos. Surge também um quadrado branco logo abaixo da palavra "Ok". Ele é chamado de "cursor" e sua posição na tela indica o local da próxima letra que você digitar.

Quanto ao teclado, parece o de uma máquina de escrever. No entanto, ele contém algumas teclas especiais que se fazem necessárias para que haja uma comunicação efetiva entre você e o computador. Estas teclas encontram-se sombreadas na figura 1.1 e suas funções são as seguintes:

Na descrição a seguir colocaremos o nome dado às teclas especiais no HOTBIT e no EXPERT nesta ordem: HOTBIT/EXPERT.

(1) SHIFT/SHIFT : Essa tecla é usada para digitar letras maiúsculas ou para obter-se os caracteres impressos na parte superior das teclas, como numa máquina de datilografia. Quando pressionada juntamente com as teclas 3 (GRAPH/LGRA) ou 4 (CODE/RGRA) obtém-se outros grupos de caracteres e símbolos gráficos, todos eles mostrados no apêndice A.

Experimente digitar algumas letras e algarismos sem pressionar e pressionando o SHIFT para ver a diferença. Se você mantiver uma tecla pressionada por alguns momentos, o processo de repetição automática será ativado.

(2) CAPS/CAPS LOCK: Cada vez que essa tecla for pressionada haverá mudança na apresentação das letras: se estiverem sendo mostradas em minúsculas passam a ser maiúsculas e vice-versa. O HOTBIT possui uma lâmpada que, se estiver acesa, indica o modo "maiúsculas". Esta tecla não influi sobre os caracteres impressos na parte superior das teclas.

(3) GRAPH/LGRA ou (4) CODE/RGRA: Mantendo-se uma dessas teclas pressionadas ao digitar-se outras, obtemos a apresentação, na tela, dos símbolos gráficos apresentados no apêndice A.

Brinque um pouco pressionando uma dessas duas teclas e, sem soltá-la, digitando letras quaisquer do teclado. Repita a experiência mantendo pressionada uma dessas duas teclas junto com a tecla SHIFT (1).

Figura 1.1A - Teclas Especiais do HOTBIT.

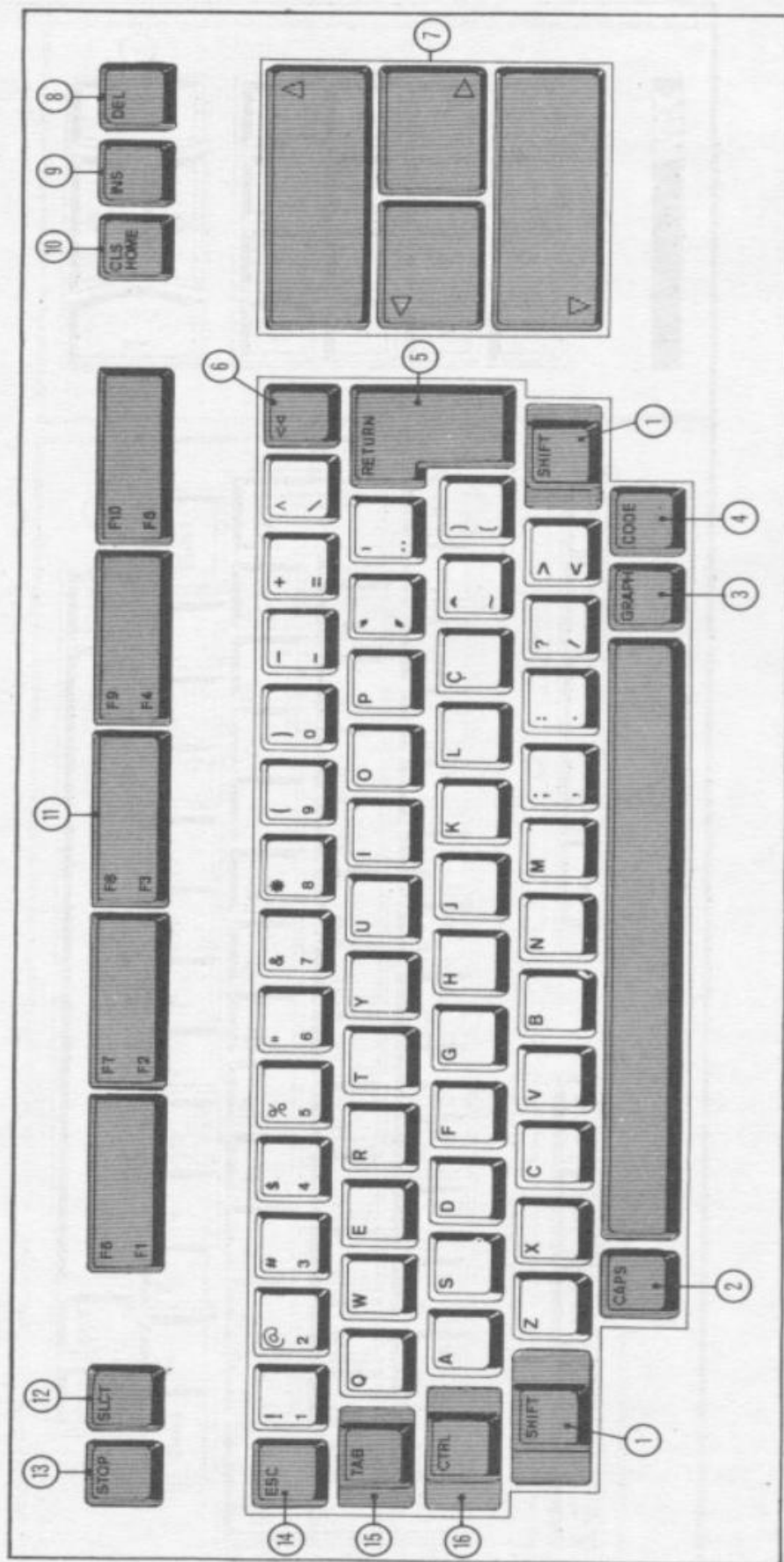
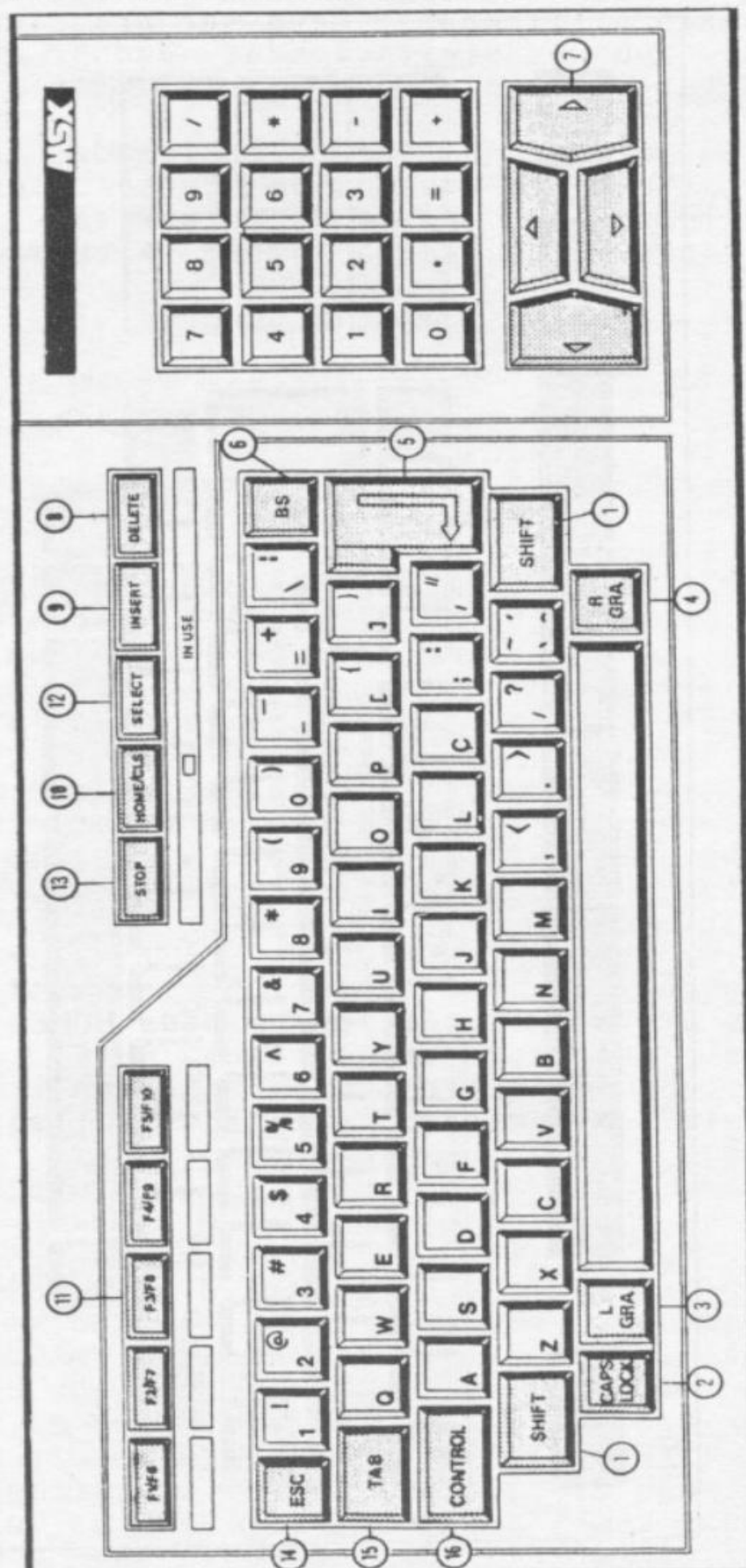



Figura 1.1 B - Teclas Especiais do EXPERT.

(5) RETURN : Essa tecla é uma das mais importantes e mais utilizadas. Deve ser pressionada invariavelmente após a digitação dos dados, comandos ou instruções para que sejam enviados à memória do computador.

(6) ◀◀/BS: Ao ser pressionada, cancela o caractere situado à esquerda do cursor. Todos os caracteres situados à direita do cursor são deslocados para a esquerda. Essa tecla também é chamada de BACK SPACE.

Experimente apagar algo que você ja colocou na tela.

(7) SETAS : Essas teclas movem o cursor nas direções e sentidos indicados. Não são afetados os caracteres já digitados.

Movimente um pouco o cursor colocando em cima de alguma letra que você já tenha digitado e apagando-a com o BACK SPACE.

(8) DEL/DELETE : Cancela o caractere sobre o qual se encontra o cursor. Todos os caracteres seguintes são deslocados de uma posição à esquerda.

Movimente o cursor com as setas e apague algumas letras com essas teclas. Perceba a diferença entre essa tecla e a do BACK SPACE.

(9) INS/INSERT : Essa tecla deve ser utilizada quando você quiser acrescentar caracteres dentro de uma linha. Mova o cursor para a posição posterior àquela em que você deseja acrescentar algo, pressione INS e digite o texto a ser inserido.

(10) CLS.HOME/HOME.CLS : Pressionando essa tecla, o cursor deve se posicionar no canto superior esquerdo da tela (posição denominada HOME). Se for pressionada juntamente com a tecla SHIFT, obtém-se a função CLS que significa "CLear Screen", ou seja, "Limpe a Tela". Nesse caso, a tela será limpa e o cursor deslocado para a posição HOME.

Faça isso para poder apagar totalmente toda "sujeira" que você fez até agora.

(11) F1...F10 : Pressionar uma dessas teclas equivale a introduzir o comando a ela correspondente. Elas existem para reduzir o trabalho de digitação pois ao pressioná-las, o computador executa o comando nelas programado evitando-se, desta forma, que você tenha que digitar cada um dos caracteres que os compõe. As funções F1 até F5 são acionadas diretamente através da simples pressão de cada uma delas. As teclas F6 a F10 são acionadas com a pressão simultânea da tecla SHIFT.

Na parte inferior do vídeo são visualizados

os comandos relativos a F1 até F5. Se pressionarmos SHIFT serão visualizados os relativos a F6 até F10.

Mexer com essas teclas agora gera um monte de coisas estranhas na tela. De qualquer forma brinque um pouco com elas para satisfazer sua curiosidade. Mais tarde aprenderemos como usá-las.

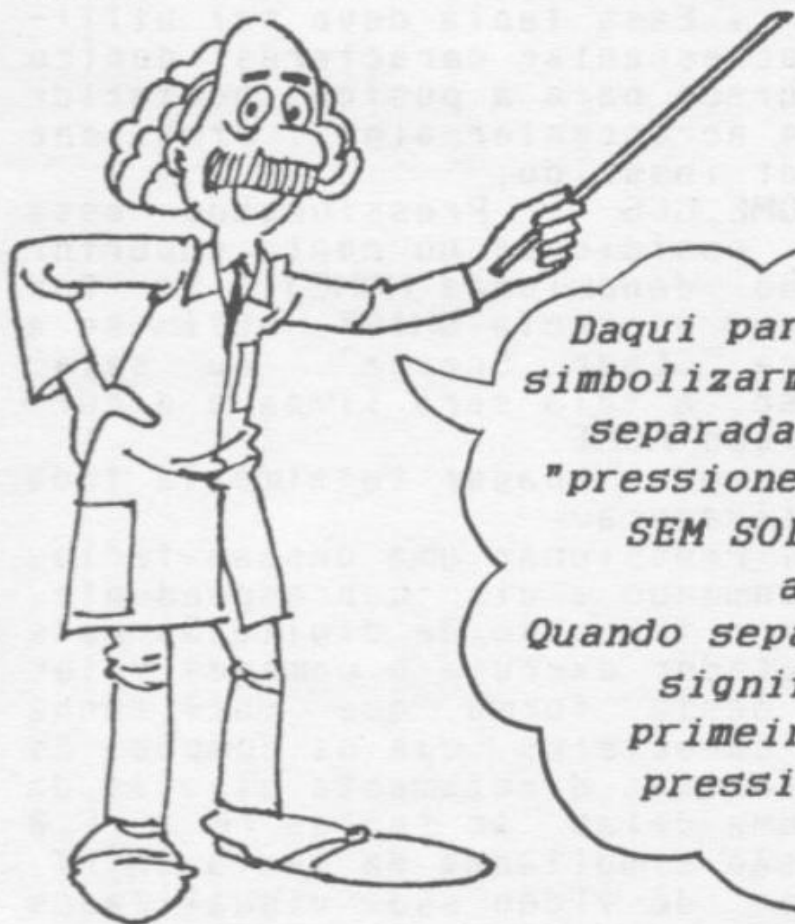
(12) SLCT/SELECT : Não é usada para programação BASIC.

(13) ESC/ESC : Essa tecla é muitas vezes usada em aplicações de software. Não tem qualquer efeito se utilizada diretamente.

(14) TAB/TAB : Pressionando essa tecla, o cursor se deslocará para a direita apagando os caracteres sobre os quais ele passar. O cursor só pára ao encontrar uma coluna múltipla de 8.

Usando as setas leve o cursor acima da sujeira que você fez agora pouco e tente apagá-la com o TAB.

(15) CTRL/CONTROL : Essa tecla, quando pressionada junto com outras, serve para acessar funções especiais.

Experimente, por exemplo, levar o cursor no meio de uma sequência de caracteres da tela e, pressionando essa tecla, digite E.

EXERCÍCIOS

1.1- Limpe a tela pressionando:

[SHIFT] + [HOME/CLS]

e digite esta sequência de caracteres:

ABCDEFGHIJKLMNOPQ■

Como eles são todos maíusculos, ative o CAPS (CAPS LOCK) pressionando uma única vez a tecla (2).

Posicione o cursor (usando as setas) sobre a letra H:

ABCDEFG■IJKLMNOPQ

1.2- Pressione a tecla DEL/DELETE (8). Você obteve:

a) ABCDEF■IJKLMNOPQ
b) ABCDEFGHIJKLMNOPQ■
c) ABCDEFG■JKLMNOPQ
d) ABCDEFG■
e) ABCDEFG ■
f) todos os caracteres da linha desaparecem.

1.3- Repita o exercício 1.1 até refazer a sequência:

ABCDEFGHIJKLMNOPQ■

com o cursor sobre o H.

Pressione agora a tecla ◀◀/BS (6). Você obteve:

a) ABCDEF■IJKLMNOPQ
b) ABCDEFGHIJKLMNOPQ■
c) ABCDEFG■JKLMNOPQ
d) ABCDEFG■
e) ABCDEFG ■
f) todos os caracteres da linha desaparecem.

1.4- Mais uma vez, refaça a sequência do exercício 1.1 com o cursor em H. Se você tiver preguiça tente usar a tecla INS/INSERT (9). Se não conseguir, não se preocupe: apague tudo e escreva de novo. Lembre-se: os dois maiores incentivadores do pro-

gresso da humanidade são a curiosidade e a preguiça!

Pressione agora as teclas:

HOTBIT : [CTRL] + [E]
EXPERT : [CONTROL] + [E]

Você deve obter:

a) ABCDEF█IJKLMNOPQ
b) ABCDEFGHIJKLMNOPQ█
c) ABCDEFG█JKLMNOPQ
d) ABCDEFG█
e) ABCDEFG █
f) todos os caracteres da linha desaparecem.

1.5- Repita o exercício 1.1 obtendo outra vez a sequência completa com o cursor em H.

Pressione as teclas:

HOTBIT : [CTRL] + [U]
EXPERT : [CONTROL] + [U]

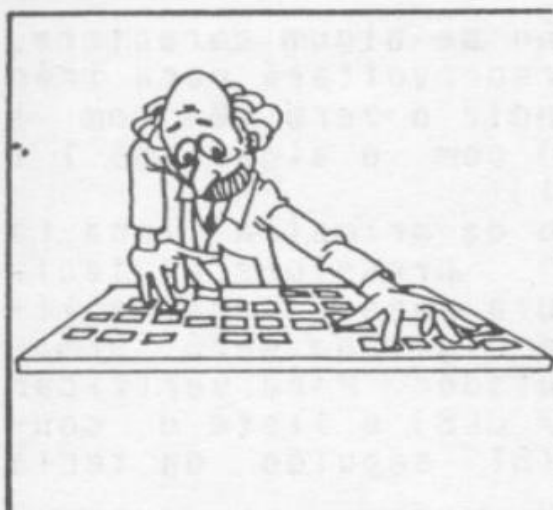
Você deve obter:

a) ABCDEF█IJKLMNOPQ
b) ABCDEFGHIJKLMNOPQ█
c) ABCDEFG█JKLMNOPQ
d) ABCDEFG█
e) ABCDEFG █
f) todos os caracteres da linha desaparecem.

1.6- Pela última vez (ufa!) repita a sequência do exercício 1.1 com o cursor em H e aperte TAB (14) duas vezes:

[TAB], [TAB]

Você obteve:

a) ABCDEF█IJKLMNOPQ
b) ABCDEFGHIJKLMNOPQ█
c) ABCDEFG█JKLMNOPQ
d) ABCDEFG█
e) ABCDEFG █
f) todos os caracteres da linha desaparecem.

AULA 2

Assunto: **Digitação de programas.**

Objetivo: **Aprender a digitar linhas de um programa e editá-las.**

DIGITAÇÃO E EDIÇÃO

Quando introduzimos, via teclado, um programa no computador, estamos lhe fornecendo, na realidade, uma sequência de instruções que mais tarde ele executará.

Para que ele saiba qual a ordem de execução, essas instruções são numeradas.

Nesta aula vamos aprender a digitar (e executar) um curto programa em BASIC, mostrado na figura 2.1 . Não se preocupe, por enquanto, em entender como ele funciona, pois o que importa agora é aprender a digitar.

Figura 2.1 - Programa Exemplo

```
10 COLOR 15,1,1
20 SCREEN 2
30 FOR I=1 TO 20
40 X=RND(1)*256
50 Y=RND(1)*192
60 R=RND(1)*40
70 CIRCLE (X,Y),R,RND(1)*15
80 NEXT I
90 GOTO 90
```

Inicialmente, aperte uma única vez a tecla CAPS/CAPS LOCK. Isso faz com que as letras sejam mostradas em maiúsculo, assim como a listagem mostrada na figura 2.1.

Comece digitando a primeira linha e observe que o cursor vai se movendo para a direita enquanto os caracteres digitados vão sendo deixados para trás.

Se você errar a digitação de algum caractere, basta apertar BACKSPACE e o cursor voltará para trás apagando. Cuidado para não confundir o zero (Ø) com a letra O, a letra L minúsculo (l) com o algarismo 1 e este com a letra i maiúscula (I)!

Ao terminar a digitação da primeira linha (a que leva o número 1Ø na frente), pressione a tecla RETURN. Desta forma, o cursor pula para a próxima linha da tela e a linha do programa digitada será armazenada na memória do microcomputador. Para verificar isso, limpe a tela (SHIFT + HOME/ CLS) e liste o conteúdo da memória comandando LIST seguido da tecla RETURN.

A tela deverá ficar como o indicado na figura 2.2.

Figura 2.2

```
LIST
10 COLOR 15,1,1
Ok
■
```

Continuando a digitação do programa, vamos novamente limpar a tela (SHIFT + HOME/CLS) e digitar a linha de número 2Ø.

Não se esqueça de teclar RETURN ao chegar ao fim da linha digitada.

Para verificar o conteúdo da memória, limpe novamente a tela (SHIFT + HOME/CLS) e liste o programa com o comando LIST (seguido de RETURN). A tela deverá agora se mostrar conforme a figura 2.3.

Figura 2.3

```
LIST
10 COLOR 15,1,1
20 SCREEN 2
Ok
█
```

Antes de continuar, podemos aprender um pequeno truque que facilitará o seu trabalho toda vez que você quiser limpar a tela e a seguir listar o programa. Como essa operação é muito repetitiva é conveniente atribuí-la a uma tecla de função.

Para programar uma tecla de função, devemos comandar:

`KEY n , "(conteúdo do comando)"`

onde n é o número da tecla que você deseja programar (1 a 10).

Se você quiser que o comando armazenado na tecla seja automaticamente executado sem que a tecla RETURN seja pressionada, então comande :

KEY n , "(conteúdo do comando)" + CHR$(13)

Por exemplo, vamos programar a tecla F1 substituindo o comando COLOR nela existente por CLS+LIST+RETURN. Assim digite :

```
KEY 1 , "CLS:LIST" + CHR$(13)
```

Após pressionar RETURN a listagem das palavras armazenadas nas teclas de função, que aparece na última linha da tela, é alterada. Observe que a palavra "color" foi substituída por "CLS:LIST".

Experimente agora apertar a tecla F1 e veja o resultado. A tela é limpa e o programa é listado sem a necessidade de pressionarmos RETURN. Isso acontece porque o código do caractere relativo à tecla RETURN também foi armazenado na programação da tecla F1 (CHR$(13)).

Digite agora a linha 30 (não esqueça o RETURN no final !). Aperte a tecla F1 e a tela terá o aspecto da figura 2.4.

Figura 2.4

```
LIST
10 COLOR 15,1,1
20 SCREEN 2
30 FOR I=1 TO 20
Ok
■
```

Vamos supor agora que você queira alterar a linha 30 para :

```
30 FOR I = 1 TO 50
```

Para corrigir a instrução que já está armazenada na memória, temos 2 soluções :

a) Redigitar toda a nova linha e, neste caso, ao teclarmos RETURN, substituimos a antiga linha 30 pela nova.

b) Editar a linha que já está na tela. Neste caso, leve o cursor até o dígito que queremos corrigir

com o auxílio das teclas de setas. Tecle o 5 em cima do 2 e pressione RETURN. Verifique este procedimento na figura 2.5.

Figura 2.5

```
* Pressione [▶] e [▲] até obter:
  30 FOR I=1 TO [2]0
* Pressione [5] e obtenha:
  30 FOR I=1 TO 5[0]
* Tecle RETURN e obtenha:
  30 FOR I=1 TO 50
  [O]k
```

Aperte agora F1 e note que a mudança foi executada.

Digamos agora que você queira alterar a linha

30 FOR I = 1 TO 50

para

30 FOR I = 1 TO 520

Neste caso, você deve levar o cursor até o 0 do número 50 e teclar INS (INSert). O cursor ficará pela metade, indicando que o próximo caractere a ser digitado não se sobreporá ao 0. Digite o 2 que você quer inserir e note que o 0 é deslocado para a direita. Tecle, a seguir, RETURN.

Acompanhe o procedimento na figura 2.6.

Figura 2.6

```
* Pressione [▶] e [▲] até obter:
  30 FOR I=1 TO 5[0]
* Tecle [INS] (INSERT) e obtenha:
  30 FOR I=1 TO 50
* Digite [2] e obtenha:
  30 FOR I=1 TO 520
* Tecle RETURN e obtenha:
  30 FOR I=1 TO 520
  [O]k
```

Tecle novamente F1 e perceba a mudança efetuada.

Vamos agora restabelecer a instrução original. Leve o cursor até o caractere 5 e digite DEL

(Delete) e depois RETURN. Veja a figura 2.7.

Figura 2.7

```
* Pressione [▶] e [▲] até obter:
  30 FOR I=1 TO [5]20
* Pressione [DEL] (DELETE) e obtenha:
  30 FOR I=1 TO [2]0
* Tecle RETURN e obtenha:
  30 FOR I=1 TO 20
  [O]k
```

Aperte F1 e continue a digitação do programa com a linha :

```
40 X=RND(1)*256
```

Não se esqueça do RETURN !

Olhando para o programa original (figura 2.1) notamos que as linhas 50 e 60 são muito parecidas com a 40. Para poupar trabalho de digitação podemos usar a linha 40 para gerar as outras duas. Para isso, desloque o cursor para o 4 e digite 5. A seguir, desloque o cursor para o X e digite Y e finalmente digite o número 192 sobre o número 256. Ao teclarmos RETURN esta nova linha é incorporada na memória sem que a 40 seja apagada. Veja a sequência das operações descritas na figura 2.8 .

Figura 2.8

```
* Use a tecla [▲] para obter:
  [4]0 X=RND(1)*256
* Digite [5] e obtenha:
  5[0] X=RND(1)*256
* Use a tecla [▶] até obter:
  50 [X]=RND(1)*256
* Digite [Y] e obtenha:
  50 Y[=]RND(1)*256
* Use a tecla [▶] até obter:
  50 Y=RND(1)*[2]56
* Digite [1], [9] e [2] e obtenha:
  50 Y=RND(1)*192■
* Tecle RETURN
```

Para se certificar que a nova linha foi armazenada, limpe a tela (SHIFT + HOME/CLS) e comande LIST 40-50. Você deverá obter na tela a listagem das

linhas 40 e 50.

Você pode agora gerar a linha 60 usando a de número 50 conforme a figura 2.9 .

Figura 2.9

* Use a tecla [▲] para obter:
  [5]0 Y=RND(1)*192
* Digite [6] e depois [▶] até obter:
  60 [Y]=RND(1)*192
* Digite [R] e depois [▶] até obter:
  60 R=RND(1)*[1]92
* Digite [4], [0] e [DEL] (DELETE) para obter:
  60 R=RND(1)*40■
* Tecle RETURN

Aperte agora a tecla F1 e depois digite as linhas 70, 80 e 90, não se esquecendo de pressionar RETURN após a digitação de cada uma delas.

Não se preocupe com a ordem com a qual as linhas são inseridas na memória. Se você introduzir a de número 90 e depois as linhas 70 e 80, o MSX se encarregará de arquivá-las em ordem numérica crescente.

Ao terminar a digitação aperte a tecla F1 para apagar a tela e listar o programa. A tela deverá mostrar-se idêntica à figura 2.1. Confira com cuidado, pois qualquer caractere trocado pode fazer com que o programa não funcione conforme o esperado ou simplesmente não funcione.

Esse cuidado é essencial em qualquer linguagem. Note, por exemplo, a diferença no significado das duas frases a seguir:

"Matar o rei não; é crime!"
"Matar o rei não é crime!"

Imagine o que acontece no caso de enviarmos caracteres errados ao computador. Ele "bagunçará" todo o programa ou se "recusará" a executá-lo porque não "entendeu" a instrução dada. Lembre-se, o computador não passa de um "burro rápido".

Dessa forma, se houver necessidade de efetuar correções no programa digitado, use tudo o que foi aprendido até aqui e nunca se esqueça de pressionar RETURN para inserir uma nova linha na memória.

Limpe a tela (SHIFT + HOME/CLS) e execute o programa comandando RUN (seguido da tecla RETURN) ou pressionando a tecla F5.

A tela deve se por no modo gráfico e devem ser desenhados vários círculos de tamanhos, cores e em posições diferentes. Para parar o programa e voltar a listá-lo comande CTRL+STOP e depois F1.

Quando você desligar o computador, o conteúdo da memória em que são escritos os programas em BASIC (RAM) é apagado. Isso acontece também se for executado o comando NEW. Desta forma, se você quiser preservar o programa digitado ele deve ser gravado.

A gravação pode ser feita em fita cassete através do comando CSAVE. Posteriormente, a recuperação (leitura) será feita com o comando CLOAD. Para saber como utilizar o gravador para arquivar programas, leia o Apêndice B.

EXERCÍCIOS

2.1- Qualquer computador MSX já vem de fábrica com 256 caracteres diferentes (numerados de 0 a 255). Digite com todo cuidado o programinha da figura 2.10 que permite ver, ampliado, qualquer caractere com seu código à frente.

Figura 2.10

```
10 OPEN"GRP:" AS # 1
20 FOR I=0 TO 255
30 SCREEN 3
40 PRINT #1,I;CHR$(I)
50 FOR T=0 TO 300
60 NEXT T
70 NEXT I
```

Rode o programa (pressionando F5) e baseado no que você vê ele fazer, complete a tabelinha a seguir:

Código		34		99	
Caractere	!		K		ü

2.2- Você deve ter notado que o micro demora um certo tempo entre um caractere e outro. Este tempo é regulado pela linha 50 do programinha da figura 2.10. Tente mudá-la para:

```
50 FOR T=0 TO 600
```

para ter o dobro do tempo entre um caractere e outro.

Para isso liste o programa (digitando LIST e depois pressionando RETURN), leve o cursor até o 3 do 300 e substitua-o por um 6.

Agora cuidado: qualquer alteração em qualquer linha do programa só é inserida na memória após pressionarmos a tecla RETURN.

Feita a alteração, rode o programa (digitando F5).

Qual é o efeito ?

2.3- Todo bom programador de computador deve ser, antes de mais nada, um bom observador. Você já reparou que, quando o programa da figura 2.10 numera os caracteres de 0 a 32, nenhum símbolo ou letra aparece?

Existe uma maneira de se corrigir isto alterando-se uma linha do programa.

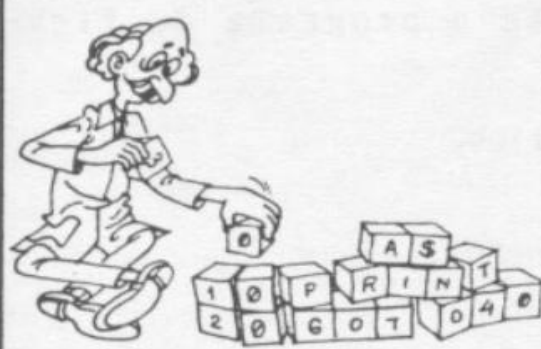
Use tudo que você aprendeu nesta aula para fazer esta alteração. A linha 40 passa a ser:

```
40 IF I < 33 THEN PRINT #1,I;CHR$(1)+CHR
$(64+I) ELSE PRINT #1,I;CHR$(I)
```

a) O que acontece agora ao rodar o programa?

b) Qual é o caractere correspondente ao código 8?

2.4- Experimente rodar o programa do exercício anterior acrescentando, antes, a linha:

```
80 GOTO 20
```

O que aconteceu ?

2.5- Com relação ao exercício 2.4, qual a melhor ma-

neira de se interromper a execução do programa para poder listá-lo outra vez ?

a) Metralhar o micro.
b) Pressionar a tecla STOP.
c) Pressionar simultaneamente as teclas CTRL+STOP.
d) Cortar o fio da tomada.
e) Desligar o computador.

AULA 3

Assunto : Primeiros passos na programação.

Objetivo : Começar a introduzir o conceito de programação.

PROGRAMAÇÃO

O computador executa as instruções a ele fornecidas de duas formas :

I) MODO DIRETO ou IMEDIATO : a instrução é executada assim que for digitada e seguida da pressão da tecla RETURN. Nesse modo, uma linha de instruções não é precedida de um número.

Por exemplo, digite :

```
PRINT"ESTE COMANDO IMPRIME ALGO NA TELA"
```

Pressione RETURN e veja o resultado.

A palavra PRINT pode ser substituída por um ponto de interrogação (?). Por exemplo :

```
? CHR$(34);3*7;CHR$(34)
```

deve resultar em : "21"

Neste comando executou-se a impressão de aspas (") com o auxílio da função CHR$(34). O número que aparece entre parênteses é o código do caractere desejado (veja o exercício 2.1).

II) MODO INDIRETO ou PROGRAMA : a instrução não é executada imediatamente após sua introdução com a tecla RETURN. Ela é armazenada na memória para posterior execução.Neste modo, as linhas de intruções devem ser precedidas de um número.

Uma sequência de instruções contidas em linhas numeradas na ordem em que devem ser executadas é

o que chamamos de programa.

Para exemplificar, digite o programa da figura 3.1.

Figura 3.1 - Exemplo de um programa simples.

```
10 CLS
20 PRINT"A RAIZ QUADRADA DE 25 É";
30 PRINT SQR(25)
40 PRINT
50 PRINT"FIM"
```

Os números das linhas devem estar compreendidos entre 0 e 65529. Eles normalmente estão em intervalos de 10 para facilitar futuros acréscimos de novas linhas.

Para executar o programa da figura 3.1, pressione a tecla F5.

Seu funcionamento é simples :

Linha 10 : limpa a tela com o comando CLS.

Linha 20 : imprime na tela o texto "A RAIZ QUADRADA DE 25 É". O ponto-e-vírgula (;) colocado no final da linha, instrui o computador a realizar a próxima impressão imediatamente após esta.

Linha 30 : imprime a raiz quadrada de 25 (SQR (25)="SQuare Root" de 25).

Linha 40 : imprime uma linha em branco. Tem o efeito de "pular" uma linha.

Linha 50 : imprime a palavra FIM.

Se você quiser apagar ou eliminar alguma linha deste programa, isso pode ser feito de dois modos: um é digitar o número da linha a ser apagada e depois teclar RETURN. O outro é usar o comando DELETE, como por exemplo :

DELETE 20 : apaga a linha 20.
DELETE 30-50 : apaga da linha 30 até a 50.
DELETE -30 : apaga desde a primeira linha do programa até a de número 30, inclusive.

Se você quiser acrescentar qualquer linha basta digitá-la e depois pressionar RETURN. Atribua a ela um número contido no intervalo definido pelos números das duas linhas entre as quais você quer inserí-la. Por exemplo, para incluir uma linha entre as de número 40 e 50 digite :

```
45 PRINT "MSX"
```

Há também a possibilidade de colocarmos mais de uma instrução em uma única linha. Para isso, separe as instruções por dois pontos (:) e respeite um número máximo de 255 caracteres por linha. Por exemplo :

```
45 PRINT "MSX" : PRINT "MEMÓRIA DISPONÍV
EL="; FRE(0);"BYTES" : PRINT
```

AS TELAS DE TEXTO

Existem dois tipos de telas no MSX que permitem escrever textos. A escolha de cada uma delas é feita através do comando SCREEN.

Se comandarmos SCREEN 0, selecionamos uma tela de 24 linhas de texto com, no máximo, 40 caracteres cada. Esta é a tela que o computador apresenta assim que é ligado.

Se selecionarmos a outra tela de textos com o comando SCREEN 1, obtemos uma tela que permite introduzir 24 linhas de texto com no máximo 32 caracteres em cada uma.

O número de caracteres por linha pode ser alterado nas duas telas com o comando WIDTH. Seu formato é :

WIDTH n

onde, n=número de caracteres por linha. Deve estar compreendido entre 1 e 40 na SCREEN 0 e entre 1 e 32 na SCREEN 1.

Por exemplo, comande :

SCREEN 0 e depois WIDTH 20

Note que qualquer texto digitado a partir de agora terá no máximo 20 caracteres por linha. Para retornar ao estado inicial da SCREEN 0 comande WIDTH 40 e da SCREEN 1 execute WIDTH 29.

Se o seu televisor estiver "comendo" as margens laterais da tela, convém comandar um número menor para o WIDTH de maneira a poder visualizar tudo que está sendo escrito.

Uma das diferenças entre a SCREEN 0 e a SCREEN 1 é que na primeira os caracteres gráficos são "truncados" e na segunda eles são impressos "por in-

teiro". Por exemplo, comande SCREEN 0 e digite as teclas GRAPH/LGRA e D simultâneamente. A seguir selecione a SCREEN 1 e volte a digitar GRAPH/LGRA e D. Note a diferença.

Note também que ao selecionarmos qualquer SCREEN a tela é previamente limpa. Isso ocorre porque é impossível a coexistência de dois tipos diferentes de telas no BASIC-MSX.

Quando manipulamos textos, um dos comandos mais utilizados é o PRINT. Ele pode vir acompanhado de alguns sinais de pontuação ou de outros comandos que possibilitam organizar a exibição dos textos na tela.

Para exemplificar, digite o programa da figura 3.2.

Figura 3.2 - O uso do comando PRINT.

```
10 KEY OFF
20 SCREEN 0 : WIDTH 30
30 PRINT"MICROCOMPUTADOR MSX"
40 PRINT
50 PRINT TAB(5)"EDITORA"TAB(20)"ALEPH"
60 LOCATE 2,6
70 PRINT"TEXTO POSICIONADO POR LOCATE"
80 PRINT
90 PRINT"ABC","123"
100 PRINT"ABC";"123"
110 PRINT
120 PRINT"TEXTO SEM"
130 PRINT"PONTO-E-VÍRGULA"
140 PRINT
150 PRINT"TEXTO COM ";
160 PRINT"PONTO-E-VÍRGULA"
```

A linha 10 do programa utiliza o comando KEYOFF que simplesmente "desliga" a visualização na parte inferior da tela dos comandos associados às teclas de função. Para visualizá-los novamente utilize KEYON.

A linha 20 seleciona a SCREEN 0 e define uma largura de 30 caracteres para o texto com o comando WIDTH.

A linha 30 efetua uma impressão a partir da primeira posição da tela.

A linha 40 tem o efeito de "pular" uma linha de texto na tela.

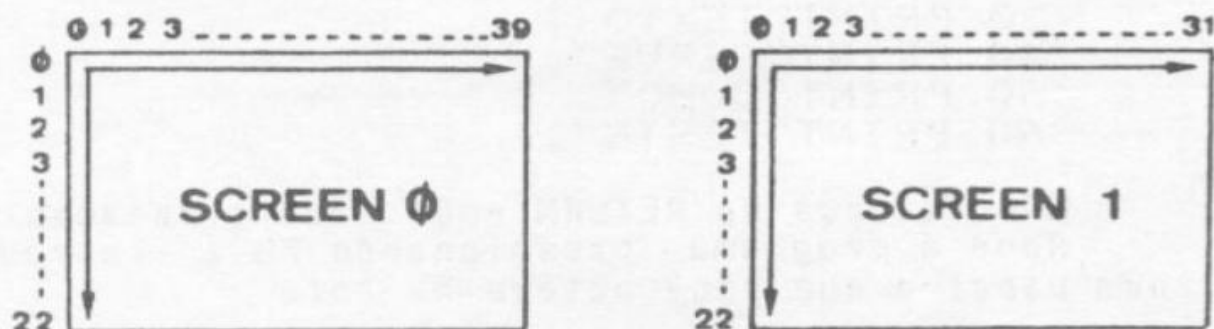
A linha 50 efetua uma impressão a partir da coluna 5 e outra a partir da coluna 20 contadas desde a margem esquerda da tela. Ela utiliza a função TAB (TABular), análoga ao tabulador de uma máquina de escrever.

A linha 60 utiliza o comando LOCATE que tem por função posicionar a próxima impressão. Sua sintaxe é :

LOCATE a,b

onde, a = número da coluna (0 a 39).
b = número da linha (0 a 22).

As coordenadas das telas de texto são definidas de acordo com a figura 3.3.

Figura 3.3 - Coordenadas das Telas de Texto.

As linhas 90 e 100 mostram a diferença entre separar textos por vírgulas ou ponto-e-vírgulas.

As demais linhas do programa são auto-explicativas.

EXERCÍCIOS

3.1- Estude com cuidado o programa da figura 3.2. A seguir apague a memória do computador digitando

NEW (e RETURN)

Digite agora este programinha:

```
10 CLS
20 PRINT"ESTE"
30 PRINT"TEXTO"
40 PRINT"SERVE"
50 PRINT"COMO"
60 PRINT"TESTE"
```

Não esqueça de pressionar RETURN após cada linha do programa para que ela seja inserida na memória.

Digite F5 (que equivale ao RUN seguido de RETURN) e reproduza num papel o que você obteve na tela do computador.

3.2- Liste o programa do exercício 3.1 digitando F9 (SHIFT+F4) e, movendo o cursor com as setas, acrescente uma vírgula após cada linha que contém PRINT:

```
10 CLS
20 PRINT"ESTE",
30 PRINT"TEXTO",
40 PRINT"SERVE",
50 PRINT"COMO",
60 PRINT"TESTE",
```

Não esqueça do RETURN após cada alteração!

Rode o programa pressionando F5 e reproduza num papel o que você obteve na tela.

Qual o efeito da vírgula ?

3.3- Novamente liste o programa do exercício 3.2 e, usando as setas para mover o cursor, substitua todas as vírgulas por ponto-e-vírgula. Não esqueça de pressionar o RETURN após cada alteração.

Digite F5 e reproduza num papel o que você obteve na tela.

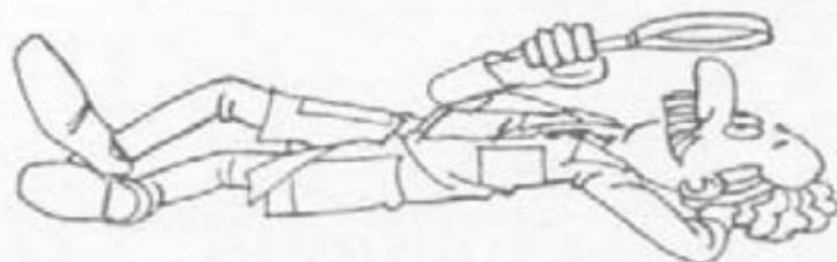
3.4- Introduza algumas linhas adicionais ao programa do exercício 3.3, de maneira que ele passe a ter o aspecto a seguir:

```
10 CLS
15 LOCATE 1,1
20 PRINT "ESTE";
25 LOCATE 3,3
30 PRINT "TEXTO";
35 LOCATE 5,5
40 PRINT"SERVE";
45 LOCATE 7,7
50 PRINT"COMO";
55 LOCATE 9,9
60 PRINT"TESTE";
```

Rode este programa com F5 e reproduza num papel o que você obteve na tela.

3.5- Altere o programa do exercício 3.4 de maneira a obter a tela da figura a seguir.

```
 0123456789012345678901234567890123456789
0
1 ESTE
2      TEXTO
3            SERVE
4       COMO
5 TESTE
6
7
```

É sempre conveniente planejar a tela desejada antes de escrever o programa que a execute.

Escreva a seguir seu programa.

```
10
15
20
25
30
35
40
45
50
55
60
```

AULA 4

Assunto: Constantes e Variáveis do MSX.

Objetivo: Estudar o tratamento que o MSX dá às constantes e às variáveis.

CONSTANTES E VARIÁVEIS

A memória do computador está capacitada a armazenar não somente linhas com instruções mas também outros elementos bastante importantes para a elaboração e execução de programas. Esses elementos são as constantes e as variáveis.

CONSTANTES

As constantes contém informações que não mudam durante a execução do programa. Elas podem ser de dois tipos : alfanuméricas ("string") ou numéricas.

As constantes alfanuméricas podem conter um máximo de 255 caracteres que devem sempre ser colocados entre aspas ("). Por exemplo, "COMPUTADOR" ou "2378". Note que apesar dos caracteres que compõem a constante "2378" serem algarismos ela não é tratada como um número e não poderá ser utilizada diretamente em cálculos.

Uma constante numérica pode ser positiva ou negativa, inteira ou não. No computador, assim como nas máquinas de calcular, a vírgula decimal deve ser substituída por um ponto. Por exemplo, o número 378,75 deve ser escrito como 378.75 (notação anglo-saxônica).

O BASIC-MSX reconhece seis tipos de constantes numéricas :

1) INTEIRA: número inteiro compreendido entre -32768 e 32767. Exemplos : 279 ou -577.

2) PONTO FIXO: número real positivo ou negativo que contém casas decimais. Exemplos: 72.7 ou -23.15.

3) PONTO FLUTUANTE: número positivo ou negativo representado na forma exponencial, análoga à notação científica. Uma constante com ponto flutuante é constituída por uma constante inteira ou de ponto fixo (positiva ou negativa) seguida da letra E (ou D) e o número de posições que o ponto decimal deve andar (se o deslocamento for para a esquerda, esse número é negativo). Exemplos : -373.97E5 = -37397000 ou
797.2E-3 = .7972

As constantes de ponto flutuante devem estar compreendidas entre 10 elevado a -64 e 10 elevado a 63.

4) HEXADECIMAL: números hexadecimais precedidos por &H. Exemplo : &HFB.

5) OCTAL: números octais identificados pelo prefixo &O. Exemplo : &O37.

6) BINÁRIA: números binários precedidos por &B. Exemplo : &B10001101.

As constantes numéricas podem também ser declaradas como números de precisão simples (6 algarismos significativos) ou dupla (14 algarismos significativos).

O computador reconhece uma constante como sendo de precisão simples quando, em ponto flutuante, utiliza-se a letra E, ou então se ela é seguida por um ponto de exclamação (!). Por exemplo :

```
378.23E9 ou
277.23!
```

A constante será tratada como de precisão dupla se, em ponto flutuante, utilizarmos a letra D ou se ela for seguida por #. Exemplos :

```
278.2D-3 ou
44.55#
```

Além disso, se nenhum sufixo, ! ou # for atribuído a uma constante ela será tratada com precisão dupla.

VARIÁVEIS

As variáveis podem ser entendidas como "caixas" que guardam informações. Essas informações podem ser alfanuméricas ou numéricas e podem ser atribuídas diretamente pelo programador ou através de cálculos

realizados pelo programa.

Para que o computador saiba localizar as informações, cada "caixa" que as guarda deve ter um nome.

O nome de uma variável pode ter qualquer tamanho. Ele pode conter letras ou números mas o primeiro caractere deve ser sempre uma letra.

Além disto, o nome da variável não pode coincidir com o de qualquer comando, função ou termo do BASIC-MSX, tais como OR, PRINT, IF, etc.

Convém notar também que o computador considera apenas os DOIS PRIMEIROS caracteres do nome de uma variável para identificá-la. Por exemplo, os nomes PREÇO e PRAZO são idênticos para o BASIC-MSX pois ambos começam com PR.

Assim como as constantes, as variáveis podem ser alfanuméricas ou numéricas.

As alfanuméricas devem sempre terminar com $ e as numéricas podem terminar com % (inteiras), ! (precisão simples) ou # (precisão dupla).

Quando o nome de uma variável não termina com qualquer um dos sufixos mencionados ($, %, ! ou #), o BASIC-MSX a considera numérica e de precisão dupla.

Exemplos :

A$ = nome de variável alfanumérica.
I% = nome de variável numérica inteira.
AZ! = nome de variável numérica de precisão simples.
PI# = nome de variável numérica de precisão dupla.
PI = nome de variável numérica de precisão dupla.

MANIPULANDO AS VARIÁVEIS

Assim que você liga o computador, todas as "caixas" que podem conter informações (variáveis) estão vazias. Se as variáveis são numéricas elas contém inicialmente zeros (0) e caso sejam alfanuméricas conterão cadeias vazias de caracteres ("").

Para "guardar" alguma informação numa variável devemos usar o comando LET. Esse comando atribui a uma variável uma informação como se a estivesse guardando na "caixa" correspondente.

Por exemplo, rode o programa da figura 4.1.

Figura 4.1

```
10 SCREEN 0
20 LET A=20
30 LET B=6
40 LET A$="multiplicado por"
50 LET P=A*B
60 LET B$="é igual a"
70 LOCATE 0,10
80 PRINT A;A$;B;B$;P
90 END
```

No programa acima foram atribuídos os valores 20 e 6 às variáveis A e B, respectivamente. Na linha 50 foi feito o produto dos dois números e o resultado foi "guardado" na variável P (o * funciona como sinal de multiplicação).

Nas linhas 40 e 60 foram atribuídas às variáveis alfanuméricas A$ e B$ sequências de caracteres (textos) que devem aparecer entre aspas (").

Se você quiser, não há necessidade de utilizar-se o comando LET para atribuir algum conteúdo a uma variável. Assim,as linhas 20 a 60 podem ser substituídas pelas que aparecem na figura 4.2.

Figura 4.2

```
20 A=20
30 B=6
40 A$="multiplicado por"
50 P=A*B
60 B$="é igual a"
```

Acrescente agora a linha :

```
25 A = 30
```

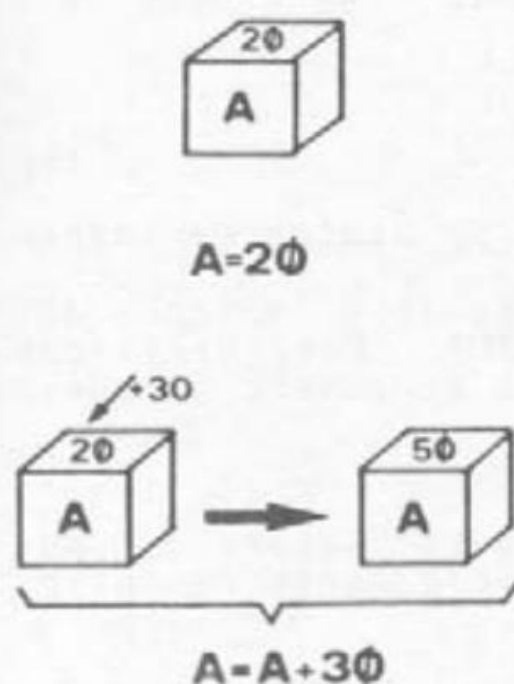
Note que a informação guardada na variável A na linha 20 (A = 20) é simplesmente substituída na linha 25 (A = 30).

Mude agora a linha 25 para :

```
25 A = A + 30
```

Pode parecer estranha, para um matemático, a instrução contida na linha 25. Para ser entendida, esta instrução não deve ser encarada como uma equação e sim como uma atribuição de um novo valor à variável A.

Ou seja, o computador executa a linha 25 tomando o valor antigo da variável A atribuído na linha 20 (A=20) e adiciona a ele mais 30 unidades. O resultado (50) é então atribuído à mesma variável A. Isso equivale a acrescentar 30 unidades às 20 originais contidas na "caixa A". Veja a figura 4.3.

Figura 4.3 - Atribuição de valores a variáveis.

Outra forma de atribuirmos uma informação a uma variável é através do comando INPUT. Sua sintaxe é:

INPUT nome de variável

Quando executamos um programa, o computador interrompe a execução ao "encontrar" o comando INPUT. Será então mostrado um ponto de interrogação e se aguardará que "alguma coisa" seja introduzida como resposta.

A resposta digitada pode ser um número ou uma sequência de caracteres que serão atribuídos à variável associada ao comando INPUT utilizado.

Para exemplificar, digite o programa da figura 4.4 .

Figura 4.4

```
10 SCREEN 0
20 INPUT"Qual o seu primeiro nome";N$
30 PRINT:PRINT"Entre dois números."
40 INPUT"Qual o primeiro";A
50 INPUT"Qual o segundo";B
60 LOCATE 0,10
70 PRINT N$;", o produto deles é";A*B
```

O comando INPUT pode ser usado também sem qualquer texto associado. Por exemplo, as linhas 40 e 50 poderiam ser digitadas como :

```
40 INPUT A
50 INPUT B
```

Nesse caso, surgem apenas os pontos de interrogação.

Além disso, poderíamos pedir a entrada dos dois valores num único comando INPUT. Para verificar isso, cancele a linha 50 e troque a de número 40 por:

```
40 INPUT A,B
```

Para entrar os valores agora, separe-os por vírgula. Por exemplo, quando surgir o ponto de interrogação, digite :

38,47 e depois tecle RETURN.

Cuidado: esta vírgula não tem nada a ver com a notação decimal. Ela apenas separa as várias entradas pedidas pelo INPUT.

EXERCÍCIOS

4.1- Associe cada uma das alternativas a seguir com cada constante relacionada:

A = alfanumérica
B = numérica inteira
C = numérica de ponto fixo
D = numérica de ponto flutuante
E = numérica hexadecimal
F = numérica octal
G = numérica binária

...(1) 312.47 ...(2) &B10011
...(3) 1.16E-19 ...(4) 2.7D4
...(5) &HF00B ...(6) "ALEPH"
...(7) "813-2033" ...(8) &O23
...(9) 0.0041 ...(10) 3

4.2- Associe cada uma das alternativas a seguir com constantes relacionadas.

A = Precisão Simples
B = Precisão Dupla

...(1) 417.49# ...(2) 2.78E9
...(3) 47.13! ...(4) 2D17
...(5) 4.31579843578

4.3- Descreva o erro cometido em cada uma das linhas a seguir:

1) "312" * 2 = 624
2) I% = 33417
3) B$ = 3
4) AZ = "MSX"
5) AZ! = 3.1D21
6) OR = 512

4.4- Como escrever a linha a seguir de maneira mais abreviada ?

```
20 PRINT A:LET B=3
```

4.5- Digite o pequeno programa a seguir:

```
10 SCREEN 1
20 R$="RAIO="
30 C$="CIRCUNFERÊNCIA="
40 A$="ÁREA="
50 PRINT R$;:INPUT R
60 PI=4*ATN(1)
70 C!=2*PI*R
80 A!=PI*R*R
90 PRINT R$;R:PRINT C$;C!:PRINT A$;A!
100 PRINT:GOTO 50
```

Rode-o e tente explicar o funcionamento das linhas 20, 30, 40, 50, 90 e 100.

4.6- Digite o programa a seguir:

```
10 SCREEN 0
20 M$="MÊS"
30 INPUT"QUAL O CAPITAL DEPOSITADO";C
40 INPUT"QUAL A TAXA DE JUROS MENSAL";T
50 T=1+T/100
60 N=N+1
70 C=C*T
80 PRINT M$;N,C
90 GOTO 60
```

Rode-o comandando RUN (e RETURN).

Quando o programa pergunta qual o capital, digite a quantidade a ser depositada na poupança (cuidado: não separe os grupos de 3 algarismos por ponto, pois o BASIC MSX interpreta isso como separação das casas decimais).

Quando o programa perguntar a taxa de juros mensal, digite-a sem o símbolo de porcentagem. Por exemplo, se a taxa de juros for 11% ao mês, digite 11 e pressione RETURN.

Veja a seguir como o capital vai "crescendo" mês após mês.

Como a tela corre muito rapidamente, pressione a tecla STOP para congelá-la. Para continuar, basta pressionar STOP novamente.

a) Use como capital o valor 2000 e taxa de juros de 14%.
   Em que mês o programa pára? Porque ?
b) Substitua o C por C% (nas linhas 30, 70 e 80). Que mudanças isto produz? Porque?
c) Agora substitua o C% por C! e responda quais foram as mudanças produzidas?

AULA 5

Assunto: Operações Matemáticas no MSX.

Objetivo: Estudar as operações aritméticas, relacionais, lógicas e funcionais.

OPERAÇÕES

O BASIC-MSX permite que seja realizada uma série de operações com constantes ou variáveis. Essas operações podem ser divididas em quatro grupos :

1) Aritméticas.
2) Relacionais.
3) Lógicas.
4) Funcionais.

Os operadores aritméticos são, em ordem de prioridade, os relacionados na Fig 5.1

Figura 5.1 - Operadores Aritméticos.

PRIORIDADE	OPERADOR	OPERAÇÃO	EXEMPLO	SIGNIFIFICADO
1	^	Exponenciação	8 ^ 4	8 elevado a 4
2	–	Mudança de Sinal	-10	-10
3	*	Multiplicação	7 * 6	7 x 6
3	/	Divisão	20/4	20 4
4	\	Divisão Inteira	10 \ 3	10 \ 3 = 3
5	MOD	Resto da Divisão	10 MOD 3	10 MOD 3 = 1
6	+	Adição	10 + 5	10 + 5
6	–	Subtração	20 - 7	20 - 7

Para mudar a prioridade (ordem de execução) devemos utilizar parênteses.

Todas as operações aritméticas só podem ser efetuadas com constantes ou variáveis numéricas, à exceção da adição, que pode também ser utilizada para "somar" variáveis alfanuméricas (concatenação de "strings"). Por exemplo :

```
A$ = "COMPUTADOR "
B$ = "MSX"
A$+B$="COMPUTADOR MSX"
```

As operações relacionais permitem comparação de constantes ou variáveis. Os operadores relacionais estão na Fig 5.2.

Figura 5.2 - Operadores Relacionais

OPERADOR	OPERAÇÃO	EXEMPLO
=	igual a	A = B
< >	diferente de	A < > B
<	menor que	A < B
>	maior que	A > B
< =	menor ou igual a	A < = B
> =	maior ou igual a	A > = B

Os operadores lógicos são usados para determinar condições múltiplas. Os principais são : OR e AND.

Para entender esses operadores imagine duas expressões que podem ser verdadeiras ou falsas. Assim:

* (expressão 1) OR (expressão 2) = VERDADEIRO se a expressão 1 ou a expressão 2 ou as duas forem verdadeiras.
* (expressão 1) OR (expressão 2) = FALSO se as duas expressões forem falsas.
* (expressão 1) AND (expressão 2)= VERDADEIRO se as duas expressões forem verdadeiras.
* (expressão 1) AND (expressão 2)= FALSO se a expressão 1 ou a expressão 2 ou as duas forem falsas.

Exemplos :

```
(5=5) OR (3=1) = VERDADEIRO
(5=6) OR ("ABC"="123") = FALSO
(1=1) AND ("ABC"="ABC") = VERDADEIRO
(7=7) AND (8=6) = FALSO
```

As operações funcionais são aquelas que utilizam as funções do BASIC MSX tais como, LOG, SIN, CHR$, etc. Há um número bastante grande delas e além disso, o usuário pode definir novas funções utilizando DEF FN. Para conhecê-las, consulte o manual do seu microcomputador.

DECISÕES

Durante a execução de um programa, muitas vezes o seu fluxo normal deve ser alterado, dependendo do resultado de algumas operações.

Para que isso seja possível, utilizamos os comandos GOTO e IF/THEN.

O comando GOTO ("vá para") provoca um desvio na execução do programa para a linha especificada por ele. Por exemplo, GOTO 100 provoca um desvio para a a linha de número 100 do programa.

O comando IF/THEN ("se/então") faz um "teste" e sua estrutura é :

IF (a condição é verdadeira) THEN (execute um comando)

Por exemplo,

```
IF A=B THEN GOTO 100
```

pode ser traduzido como : "se o conteúdo da variável A for igual ao da variável B então desvie para a linha 100 do programa".

Para ilustrar o exposto acima, analise o proprograma da figura 5.3.

Figura 5.3

```
10 CLS
20 INPUT"Entre um número";A
30 IF A MOD 2=0 THEN GOTO 60
40 PRINT"O número introduzido é ÍMPAR"
50 GOTO 20
60 PRINT"O número introduzido é PAR"
70 GOTO 20
```

Nesse programa, se o número introduzido for par então o resto de sua divisão por 2 é igual a zero

(A MOD 2 = 0). Assim, ocorrerá desvio para a linha 60. Caso o número introduzido seja ímpar, não ocorrerá tal desvio e serão executadas as linhas 40 e 50 do programa.

Execute esse programa comandando RUN e introduza alguns números. Quando cansar, comande CTRL+STOP.

Modifique agora o programa cancelando as linhas 60 e 70 e alterando a de número 30 para :

```
30 IF A MOD 2=0 THEN PRINT "O número int
roduzido é PAR" : GOTO 20
```

O funcionamento do programa não é alterado. Note que a instrução GOTO 20 introduzida na mesma linha passa a pertencer ao teste executado por IF/THEN.

Outra mudança que pode ser feita é eliminar a linha 40 e novamente alterar a de número 30 para :

```
30 IF A MOD 2=0 THEN PRINT "PAR" ELSE P
RINT "ÍMPAR"
```

A tradução dessa linha é: "SE (IF) o resto da divisão por 2 for igual a zero ENTÃO (THEN) imprima PAR, CASO CONTRÁRIO (ELSE) imprima ÍMPAR".

Digite agora o programa da figura 5.4

Figura 5.4

```
10 CLS
20 A=1
30 LOCATE A,A
40 PRINT"EDITORA ALEPH"
50 A=A+1
60 IF A<21 THEN GOTO 30
70 END
```

Este programa estabelece o que chamamos de "laço". Na linha 20 é atribuído um valor inicial à variável A (no caso, 1). Na linha 30, posiciona-se a próxima impressão de modo que tanto a coluna quanto a linha em que ela deve se iniciar correspondam ao valor de A. A linha 40 efetua a impressão. A linha 50 atribui um novo valor à variável A como sendo o seu valor atual acrescido de uma unidade. A linha 60 verifica se a variável A é menor que 21 e se for, ocorre desvio para a linha 30, "fechando o laço" e repetindo o pro-

cesso. Quando a variável A assumir o valor 21, o programa "escapa" do laço definido pelas linhas 30 a 60 e termina.

O efeito do programa acima pode também ser obtido com os comandos FOR e NEXT. Teste isso rodando o programa da figura 5.5

Figura 5.5

```
10 CLS
20 FOR A=1 TO 20
30 LOCATE A,A
40 PRINT"EDITORA ALEPH"
50 NEXT A
60 END
```

O comando FOR declara o nome da variável e de quanto a quanto ela deve variar. O comando NEXT fecha o laço. Ambos devem ser sempre utilizados conjugadamente. Não use um, sem o outro.

O incremento da variável é sempre igual a 1 a não ser que você especifique outro incremento com o comando STEP. Por exemplo, substitua a linha 20 por :

```
20 FOR A=1 TO 20 STEP 2
```

Note agora que as impressões são realizadas em linhas e colunas alternadas pois a variável A assume os valores 1,3,5,...,17 e 19, pulando de 2 em 2.

EXERCÍCIOS

5.1- Digite o programa da figura 5.6. Ele calcula a média aritimética de 5 números, escolhidos pelo usuário:

Figura 5.6

```
10 SCREEN 0
20 FOR I=1 TO 5
30 PRINT"O NúMERO";I;
40 INPUT"VALE";A
50 T=T+A
60 NEXT I
70 PRINT
80 PRINT"A MéDIA ARITMéTICA DESTES";5
90 PRINT"NúMEROS VALE";T/5
```

Rode o programa com RUN (e RETURN). Como você pode ver, a linha 10 inicializa a tela 0 (modo texto de até 40 colunas).

A linha 20 inicia um laço que vai de 1 até 5, pois decidimos tirar a média de 5 números.

As linhas 30 e 40 pedem ao usuário que digite o número 1 (que na memória do computador é armazenado na "caixa" rotulada por A).

A linha 50 pega o valor contido na "caixa" T (de total) e acresce o valor de A que acaba de ser introduzido. Obviamente o valor inicial de T é zero.

A linha 60 manda o computador para a linha 30 enquanto o I não valer 5.

Novamente o computador pede o valor do número seguinte e vai acumulando-o na "caixa" do total (T).

Quando o laço se completa (I=5), o computador imprime a mensagem e o valor da média dos 5 números, ou seja o total dividido por 5 (T/5).

Rode o programa algumas vezes para perceber o seu funcionamento.

No fim, comande LIST (e RETURN) para ver o programa na tela e responda à seguinte pergunta:

Em que linhas ele deve ser alterado para que calcule a média aritimética de 6 números?

Faça estas alterações e rode o programa para ver se funciona.

Lembre-se: toda vez que uma linha é alterada

você deve digitar RETURN em seguida para que a alteração seja comunicada à memória do micro.

5.2- Retomando o exercício anterior, vamos alterar o programa de maneira que o próprio usuário, ao iniciar a execução do programa, possa decidir a quantidade de números cuja média queremos calcular. Para isso comece acrescentando a linha:

```
15 INPUT "QUANTOS NÚMEROS";N
```

Na "caixa" N o micro armazenará a quantidade de números escolhidos pelo usuário.

Em que outras linhas devemos fazer alterações para que o programa funcione para N números?

5.3- O seu micro é capaz de gerar números ao acaso, usando a função RND. Digite o programinha a seguir:

```
10 SCREEN 0
20 FOR I=1 TO 10
30 INPUT A
40 PRINT RND (-TIME)
50 NEXT I
```

A linha 30 serve para por o computador na "espera". Ao rodar o programa, quando aparecer o ? pressione o RETURN e o computador gerará um número (na linha 40) entre 0 e 1 cujo valor depende de quanto tempo você demorou para apertar RETURN após o ponto de interrogação (?) ser colocado na tela. Rode o programa algumas vezes. A sequência obtida é sempre a mesma?

5.4- Se você quiser gerar ao acaso números INTEIROS entre 1 e 5, por exemplo, basta alterar a linha 40 do programa anterior de maneira que ele fique conforme a listagem a seguir:

```
10 SCREEN 0
20 FOR I=1 TO 10
30 INPUT A
40 PRINT INT (5*RND(-TIME))+1
50 NEXT I
```

Que alteração você faria para que o programa gere, ao acaso números inteiros entre 1 e 9 ?

5.5- Vamos aproveitar o que aprendemos até agora para criar um programa de treino de tabuada. Digite a listagem a seguir:

```
100 SCREEN 0
110 FOR I=1 TO 4
120 PRINT"APERTE RETURN"
130 INPUT T
140 A=INT(9*RND(-TIME))+1
150 PRINT"APERTE RETURN"
160 INPUT T
170 B=INT(9*RND(-TIME))+1
180 P=A*B
190 PRINT A;"X";B;"=";
200 INPUT C
210 IF C=P THEN PRINT"MUITO BEM" ELSE PR
INT"ERRADO"
220 PRINT
230 NEXT I
```

Explique o que fez a linha 210.

5.6- Com relação ao exercício anterior, que linha do programa você alteraria para que o programa, ao invés de propor 4 testes, propusesse 10 ?

5.7- Altere a linha 210 do programa de questão 5.5 para:

```
210 IF C=P THEN PRINT"MUITO BEM" ELSE GO
TO 100
```

Rode o programa assim alterado.

Qual a principal diferença entre esta versão e a anterior?

Se você fosse um programador encarregado de gerar programas para crianças, qual das duas você escolheria?

AULA 6

Assunto: Recursos Gráficos.

Objetivo: Utilizar cores, as várias telas, e as funções PSET, PRESET, LINE, CIRCLE, PAINT e DRAW.

CORES E DESENHOS

Quando ligamos o MSX a tela apresentada é a SCREEN 0 em fundo preto (azul escuro no HOTBIT) e caracteres impressos em branco. Essas cores podem ser alteradas com o comando COLOR. Sua sintaxe é a seguinte :

COLOR x,y,z

onde, x = código da cor do texto (frente).
y = código da cor da tela (fundo).
z = código da cor das bordas da tela.

Os códigos e as correspondentes cores são mostrados na figura 6.1.

Figura 6.1 - Tabela de Cores do MSX.

CÓDIGO DA COR	NOME DA COR
0	incolor
1	preto
2	verde
3	verde claro
4	azul escuro
5	azul claro
6	vermelho escuro
7	azul anil
8	vermelho
9	vermelho claro
10	amarelo ouro
11	amarelo
12	verde escuro
13	roxo
14	cinza
15	branco

Se, por exemplo, você comandar :

COLOR 1,11

deverá obter caracteres pretos em fundo amarelo.
Experimente agora :

COLOR 8,8

O texto desaparece e a tela fica toda vermelha. Se você digitar qualquer coisa, apenas ouvirá o "clic" das teclas mas os caracteres digitados não aparecerão. Isso ocorre pois é como se você estivesse tentando escrever com tinta vermelha sobre papel vermelho. Para voltar a visualizar os caracteres pressione RETURN duas ou três vezes e a seguir pressione SHIFT + F1 (F6).

O terceiro número do comando COLOR é referente à cor da borda. Ele não foi utilizado até agora pois a borda não aparece em SCREEN 0. Se, por outro lado, você estiver em SCREEN 1, podem ser especificadas as três cores : de frente, do fundo e da borda. Por exemplo, para obter-se texto em cinza sobre tela vermelha de bordas amarelas, comande :

SCREEN 1

e depois

COLOR 14,8,11

AS TELAS GRÁFICAS

Existem mais duas telas além das já mencionadas no BASIC-MSX. Elas permitem que sejam desenhados gráficos e figuras. Também devem ser selecionados pelo comando SCREEN.

Comandando SCREEN 2, obtemos a tela para desenhar em Alta Resolução Gráfica. Ela contém 192 linhas e 256 colunas.

Ao executarmos SCREEN 3, selecionamos a tela para desenhar em Baixa Resolução Gráfica. Ela dispõe de 48 linhas por 64 colunas.

As telas gráficas são desativadas toda vez que um programa termina ou é interrompido por CTRL+STOP. Nesses casos, ativa-se automaticamente a SCREEN 0.

Quando trabalhamos com SCREEN 2 ou 3, podemos contar com vários comandos específicos para traçar

pontos, linhas, círculos, podendo definir as cores de cada um.

Quanto às cores, podem ser utilizadas as 16 já mencionadas na fig. 6.1. O comando COLOR funciona da forma já explicada com uma diferença : ele pode se referir particularmente à cor de cada figura desenhada.

Para desenhar um ponto, utilize a instrução PSET. Sua sintaxe é :

PSET (x,y),c

onde, x = coordenada horizontal.
y = coordenada vertical.
c = código da cor (0 a 15)

As coordenadas são definidas na SCREEN 2 conforme a figura 6.2.

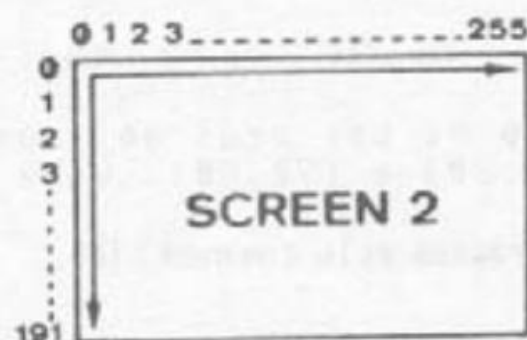
Figura 6.2 - Coordenadas da tela definida por SCREEN 2.

Digite agora o programa da figura 6.3.

Figura 6.3

```
10 SCREEN 2
20 X=RND(1)*255:Y=RND(1)*191:C=INT((RND(
1)*15)+1)
30 PSET (X,Y),C
40 GOTO 20
```

Deixe esse programa rodar por uns 10 minutos e veja o efeito obtido na tela. Note que, em SCREEN 2, as cores são definidas para cada 4 pontos horizontais consecutivos. Quando é dado um PSET num certo ponto com uma certa cor, os três pontos vizinhos a este passam a ter a mesma cor. Isso não acontece em SCREEN 3.

Para experimentar, mude a linha 10 para:

```
10 SCREEN 3
```

e verifique o resultado rodando novamente o programa.

Para apagar um ponto previamente desenhado, pode-se usar a instrução PRESET. Por exemplo, PRESET (5,10) apaga o ponto desenhado por PSET(5,10).

Quando se omite a indicação da cor, a instrução PSET assume a "de frente" que foi especificada pelo comando COLOR e a instrução PRESET, por sua vez, assume a "de fundo".

Para desenhar um segmento de reta utiliza-se a instrução LINE. Sua sintaxe é :

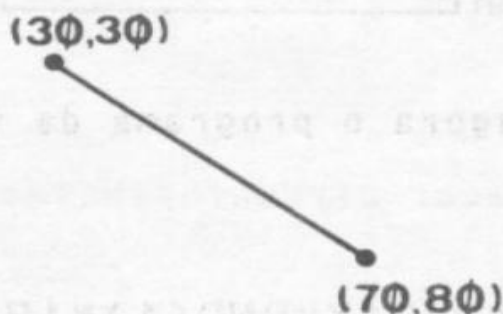
```
LINE (x1,y1)-(x2,y2),c
```

onde, (x1,y1) e (x2,y2) são as coordenadas dos dois pontos correspondentes às extremidades do segmento a ser desenhado e c é a cor desse segmento.

Por exemplo:

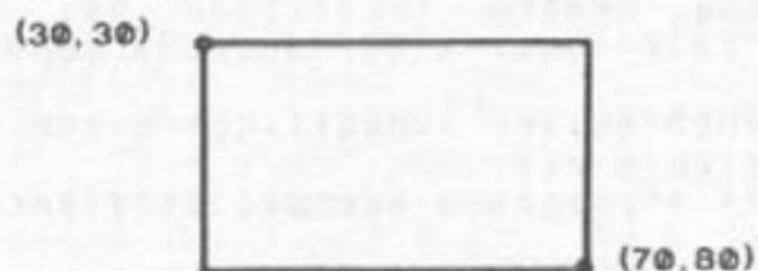
```
LINE (30,30)-(70,80),7
```

desenha um segmento na cor azul anil entre os pontos de coordenadas (30,30) e (70,80). Veja a figura 6.4.

Figura 6.4 - Linha traçada pelo comando LINE.

Para traçar um retângulo, colocamos a letra B ("Box") no fim da instrução. Nesse caso, as coordenadas indicam os vértices superior esquerdo e inferior direito do retângulo. Por exemplo:

```
LINE (30,30)-(70,80),8,B
```

desenha um retângulo vermelho conforme o indicado na figura 6.5.

Figura 6.5 - Retângulo obtido com LINE seguido da letra B.

Para colorir a parte interna do retângulo, usam-se, no fim da instrução, as letras BF ("box fill"). Por exemplo:

```
LINE (30,30)-(70,80),8,BF
```

desenha um retângulo vermelho totalmente pintado.

O programa da figura 6.6 desenha linhas em forma de grelha com as cores variando.

Figura 6.6

```
10 COLOR 15,1,7 : SCREEN 2
20 C=1
30 FOR X=0 TO 255 STEP 4
40 C=C+1
50 IF C>=15 THEN C=2
60 LINE (X,1)-(X,190),C
70 NEXT X
80 GOTO 30
```

Para traçar circunferências, elipses ou arcos, utilize a instrução CIRCLE. A sua sintaxe é :

```
CIRCLE (x,y),r,c,ai,af,a
```

onde, (x,y) = coordenadas do centro.
r = raio.
c = cor.
ai = ângulo inicial do arco (radianos).
af = ângulo final do arco (radianos).
a = relação entre o eixo vertical e o eixo horizontal.

Lembre-se que uma circunferência completa tem 6.28 radianos.

Por exemplo, querendo traçar uma circunferência com o seu centro localizado nas coordenadas (128,90) e de raio igual a 50, devemos comandar CIRCLE (128,90),50.

Se você quiser especificar a cor, acrescente o código relativo a ela.

Digite o programa exemplo da figura 6.7.

Figura 6.7

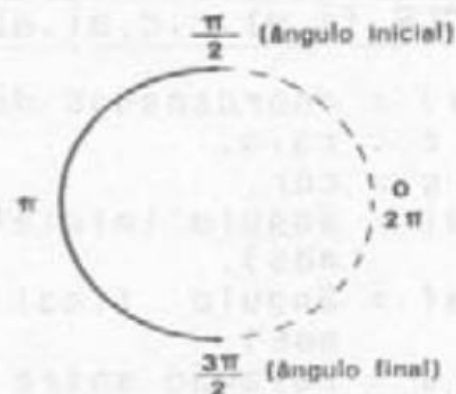
```
10 SCREEN 2 : COLOR 15,1,1
20 FOR X=10 TO 255 STEP 5
30 CIRCLE (X,80),40,8
40 NEXT X
50 GOTO 50
```

Com a instrução CIRCLE podemos traçar arcos especificando os ângulos inicial e final medidos em radianos. Para exemplificar, rode o programa da figura 6.8.

Figura 6. 8

```
10 SCREEN 2 : COLOR 15,1,1
20 PI=4*ATN(1)
30 CIRCLE (128,90),70,7,PI/2,3*PI/2
40 GOTO 40
```

Na linha 20 definimos a variável PI (3.1416) utilizando a função arco-tangente. A linha 30 traça um arco de circunferência na cor azul (7) conforme a figura 6.9.

Figura 6.9 - Arco de circunferência traçado pela instrução CIRCLE.

Se colocarmos o sinal de "menos" antes dos valores dos ângulos inicial e final, serão traçados setores circulares. Experimente substituir a linha 30 do programa da figura 6.8 por:

```
30 CIRCLE (128,90),70,7,-PI/2,-3*PI/2
```

Quando os valores dos ângulos inicial e final são omitidos, o computador assume 0 e 6.28 (2*PI), respectivamente.Desta forma é obtida a circunferência completa.

Nos exemplos citados até agora você deve ter notado que as circunferências traçadas não são exatamente "redondas"! Isso ocorre pois a "largura" é maior que a "altura" nos pontos traçados na tela. Para obtermos uma circunferência "redonda" acrescentamos mais um parâmetro para a instrução CIRCLE : o "achatamento".

O "achatamento" pode ser entendido como a razão entre os eixos vertical e horizontal. Veja a figura 6.10.

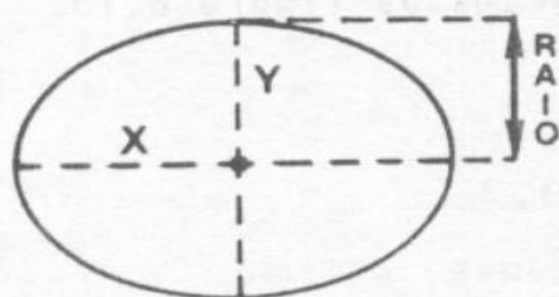
Figura 6.10 - Achatamento.

Se fizermos o "achatamento" ser igual a 1.25, obtemos a "tão desejada circunferência redonda"! Nesse caso, o raio é referente à medida do semi-eixo vertical (figura 6.10). Para verificar isso, digite o programa da figura 6.11.

Figura 6.11

```
10 SCREEN 2 : COLOR 15,1,1
20 CIRCLE (128,80),60,7,,,1.25
30 GOTO 30
```

Note na linha 20 que, quando omitimos alguns parâmetros, as vírgulas relativas a eles não devem ser

omitidas.

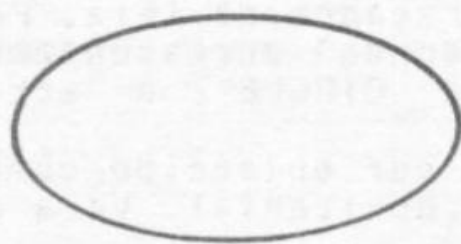
Altere agora o "achatamento" trocando a linha 20 por:

```
20 CIRCLE (128,80),60,7,,,0.8
```

ou

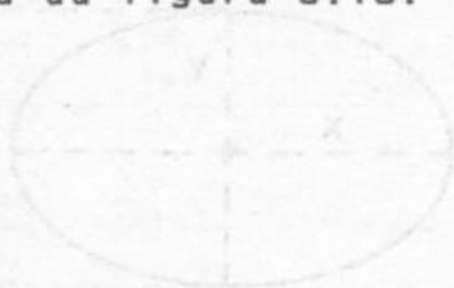
```
20 CIRCLE (128,80),60,7,,,3
```

Você deverá obter elipses conforme as da figura 6.12.

Figura 6.12 - Elipses.

ACHAT < 1.25 ACHAT > 1.25

Digite agora o programa da figura 6.13.

Figura 6.13

```
10 SCREEN 2
20 CIRCLE (128,80),50,7
30 PAINT (128,80),7
40 GOTO 40
```

O comando PAINT utilizado na linha 30 pinta a região interior da circunferência traçada pela linha 20. Sua sintaxe é:

PAINT (x,y),c

onde, (x,y) = coordenadas de um ponto qualquer pertencente à região a ser pintada.

c = cor da tinta utilizada na pintura.

O microcomputador considera a região a ser pintada como sendo aquela delimitada por uma linha fechada (fronteira) à qual pertence o ponto definido pe-

las coordenadas do comando PAINT.

Quanto à cor utilizada por PAINT, deve ser a mesma da linha que define a fronteira quando estivermos utilizando SCREEN 2.

Para desenhar quaisquer figuras, o MSX possui uma função extremamente poderosa chamada DRAW ("desenhe"). Sua sintaxe é :

DRAW string

onde, string é uma cadeia de caracteres ou uma variável alfanumérica contendo os códigos da figura 6.14.

Figura 6.14 - Algumas instruções macro-gráficas do comando DRAW.

Un	traça uma linha para cima da posição fixada para o cursor.
Dn	traça uma linha para baixo.
Ln	traça uma linha para esquerda.
Rn	traça uma linha para direita.
En	traça uma linha diagonalmente para cima e direita.
Fn	traça uma linha diagonalmente para baixo e direita.
Gn	traça uma linha diagonalmente para baixo e esquerda.
Hn	traça uma linha diagonalmente para cima e esquerda.
Cn	especifica a cor (n = 0 a 15).
Sn	especifica a escala.
B	permite movimento sem traçar qualquer linha.
N	permite traçar linha sem movimento do cursor.

Para exemplificar, o comando:

```
DRAW "R50D50L50U50"
```

deve traçar uma quadrado conforme o da figura 6.14.

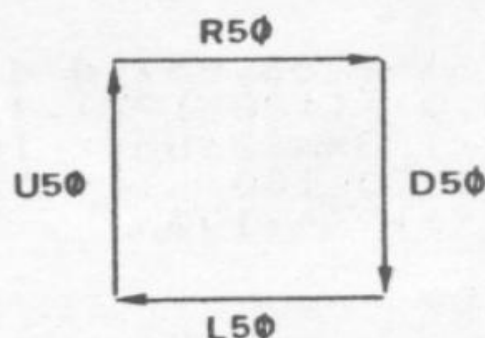
Figura 6.14 - Quadrado traçado pelo comando DRAW.

Digite agora o programa da figura 6.15 e veja o seu efeito.

Figura 6.15

```
10 SCREEN 2 : COLOR 15,1,1
20 PSET (128, 80),0
30 DRAW "C7NU70ND70NL70NR70NE70NF70NG70N
H70"
40 DRAW"BH70R140D140L140U140"
50 GOTO 50
```

EXERCÍCIOS

6.1- Elabore um programa que desenhe a bandeira japonesa na tela. Para tanto você deve prever as seguintes etapas:

1) Colocar o micro na tela gráfica de alta resolução;
2) Desenhar um retângulo cheio de branco usando o comando LINE (e a opção BF), com as seguintes coordenadas:
   canto superior esquerdo: (10,10)
   canto inferior direito: (131,111)
3) Desenhar um círculo vermelho com centro em (71,61) e raio igual a 25;
4) Pintar o interior do círculo de vermelho usando o comando PAINT;
5) Congelar o programa finalizando-o com uma linha do tipo: 100 GOTO 100

6.2- Usando o comando DRAW, desenhe na tela de alta-resolução um retângulo de largura 30 e altura 20.

6.3- Digite o programa a seguir:

```
10 SCREEN 2
20 LINE (0,96)-(256,96),4
30 LINE (128,0)-(128,192),4
40 DEF FNY (X)=3*X^2+80*X-18000
50 FOR X=-100 TO 100
60 YC=(-FNY(X)+96)/100
70 XC=X+128
80 PSET(XC,YC)
90 NEXT X
100 GOTO 100
```

A linha 10 ativa a tela gráfica de alta-resolução. As linhas 20 e 30 desenham um sistema de eixos cartesianos ortogonais com origem no centro da tela.

A linha 40 define a função do segundo grau.

$$Y = 3 X^2 + 8 X - 18000$$

escrita conforme a simbologia do BASIC MSX.

As linhas 50 e 90 determinam um laço que calcula na tela os pontos da função com o X variando de -100 a +100.

As Linhas 60 e 70 ajustam os valores de X e Y para que a origem seja deslocada para o centro da tela (lembre-se que inicialmente a origem está no canto superior esquerdo e o eixo Y está orientado para baixo.

Rode o programa e você verá uma parábola (gráfico de equação do 2o. grau) sendo desenhada na tela.

a) Altere o programa para que ele desenhe o gráfico da função

$$Y = 8 X^2 + 8 X - 18000$$

Qual o efeito que teve o aumento do coeficiente do termo de segundo grau?

b) Altere o programa para que ele desenhe o gráfico da função

$$Y = -8 X^2 - 300 X - 8000$$

Qual o efeito que teve a mudança de sinal no coeficiente do termo de segundo grau?

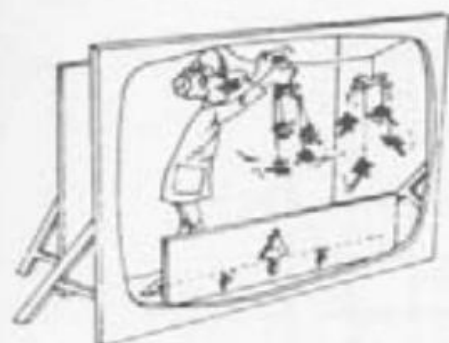
c) Que linha do programa você alteraria para desenhar os eixos em vermelho?

d) Para que serve a linha 100?

e) Mude a linha 50 para

```
50 FOR X=-100 TO 100 STEP 0.2
```

Qual o efeito que esta mudança teve sobre a "densidade" do gráfico e a rapidez do desenho?

Vamos agora estudar mais alguns recursos gráficos do MSX e ver como construir um jogo.

AULA 7

Assunto: Recursos Gráficos.

Objetivo: Aprender a imprimir texto em tela gráfica e utilizar os SPRITES.

TEXTO IMPRESSO EM TELA GRÁFICA

No modo gráfico o comando PRINT não funciona para imprimir os caracteres na tela.

Para conseguirmos visualizar os caracteres em SCREEN 2 ou 3, devemos usar as instruções :

```
OPEN "GRP:" FOR OUTPUT AS #1
```

e

```
PRINT #1, "texto a ser impresso"
```

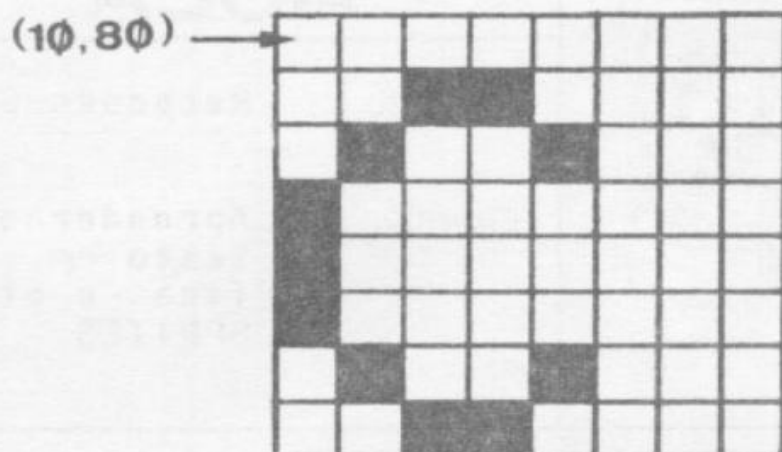
Para posicionar o início do texto na tela devemos utilizar os comandos PSET ou PRESET (e não LOCATE).

Veja o programa da figura 7.1 para ver como posicionar, imprimir e depois apagar um texto em SCREEN 2.

Figura 7.1

```
10 SCREEN 2
20 COLOR 7,1,1
30 OPEN"GRP:" FOR OUTPUT AS #1
40 PRESET(10,80)
50 PRINT #1,"COMPUTADOR"
60 FOR I=1 TO 2000
70 NEXT I
80 LINE (10,80)-(89,87),1,BF
90 GOTO 90
```

A linha 40 indica o ponto referente ao vértice superior esquerdo do campo 8x8 relativo à posição da primeira letra a ser impressa. Veja a figura 7.2.

Figura 7.2 - Primeira letra a ser impressa.

A linha 5Ø efetua a impressão e as de número 6Ø e 7Ø determinam um intervalo de tempo com um "laço vazio".

A linha 8Ø desenha um retângulo sobre a palavra COMPUTADOR e o pinta na cor de fundo (no caso, preto = 1). Veja a figura 7.3.

Figura 7.3 - Apagando um texto impresso em SCREEN 2.

MOVIMENTAÇÃO DE UMA FIGURA - SPRITE

O BASIC-MSX permite desenhar na tela figuras que podem se movimentar sem interferir nos textos ou figuras já desenhadas. Essas figuras, passíveis de animação, são chamadas de "SPRITES" e podem ser usadas em SCREEN 1, 2 ou 3.

Para utilizar os "SPRITES", devemos inicialmente defini-los, ou seja, determinar a forma das figuras a eles associadas. Apesar de um "SPRITE" poder ser definido nos formatos 8x8 e 16x16, vejamos como fazê-lo no formato 8x8.

Inicialmente faça uma figura num quadriculado 8x8 e associe a cada ponto "cheio" o valor 1 e a cada ponto "vazio" o valor Ø. Veja um exemplo na figura 7.4.

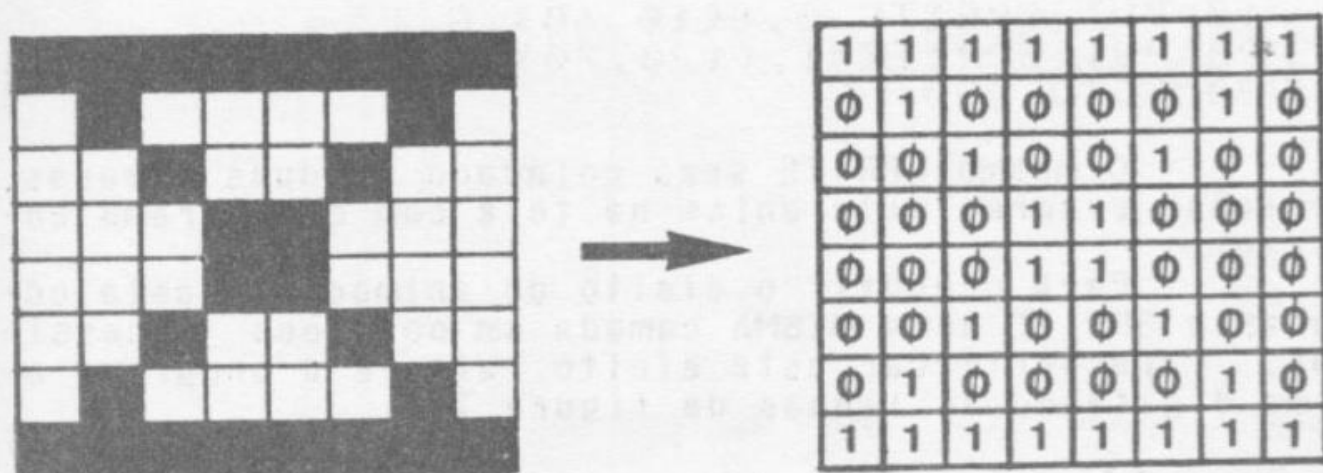
Figura 7.4 - Definindo um SPRITE.

A seguir, utilizando a variável SPRITE$ (n), onde n deve ser o número do SPRITE e estar compreendido entre Ø e 255, defina-o conforme o programa da figura 7.5.

Figura 7.5

```
100 SCREEN 2:COLOR 15,1,1
110 A$(1)=CHR$(&B11111111)
120 A$(2)=CHR$(&B01000010)
130 A$(3)=CHR$(&B00100100)
140 A$(4)=CHR$(&B00011000)
150 A$(5)=CHR$(&B00011000)
160 A$(6)=CHR$(&B00100100)
170 A$(7)=CHR$(&B01000010)
180 A$(8)=CHR$(&B11111111)
190 FOR I=1 TO 8:B$=B$+A$(I):NEXT I
200 SPRITE$(1)=B$
```

Para fazer aparecer o SPRITE definido na tela devemos usar a instrução PUT SPRITE, cuja sintaxe é :

PUT SPRITE p,(x,y),c,n

onde, p = número do plano ou camada (Ø a 31).
(x,y) = coordenadas do vértice superior esquerdo do SPRITE.
c = cor do SPRITE.
n = número do SPRITE.

No programa anterior, acrescente as linhas da figura 7.6.

Figura 7.6

```
210 PUT SPRITE 0,(110,60),8,1
220 PUT SPRITE 1,(120,70),7,1
230 GOTO 230
```

O mesmo SPRITE será colocado em duas camadas, posições e cores diferentes na tela com o programa anterior.

Para produzir o efeito de animação, basta colocar o SPRITE numa MESMA camada em posições sucessivas. Para verificar este efeito, altere o programa acima digitando as linhas da figura 7.7.

Figura 7.7

```
210 C=2
220 FOR X=10 TO 250
230 PUT SPRITE 0,(X,60),C,1
240 NEXT X
250 C=C+1:IF C>15 THEN C=2
260 GOTO 220
```

Note que quando colocamos o SPRITE numa mesma camada mas numa posição diferente, o que havia sido colocado na posição anterior é apagado automaticamente.

Resumindo, as regras que estabelecem o uso dos SPRITES são :

1) Os SPRITES podem ser colocados em SCREEN 1, 2 ou 3.

2) Não é possível colocar-se mais de um SPRITE simultaneamente numa mesma camada.

3) Podem ser utilizadas 32 camadas e portanto apenas 32 SPRITES podem aparecer simultaneamente na tela, um em cada camada.

4) É possível inserir simultaneamente numa linha horizontal da tela um máximo de 4 SPRITES, em camadas diferentes.

5) Quando se sobrepõem 2 SPRITES de diferentes camadas, o que estiver na camada de menor número tapará o outro.

EXERCÍCIOS

7.1- Digite o programa da figura 7.8 e rode-o.

Figura 7.8

```
10 SCREEN 2
20 FOR C=1 TO 11
30 READ A
40 FOR I=1 TO 10
50 LINE (10,C*10+I)-(110,C*10+I),A
60 NEXT I
70 NEXT C
80 DATA 6,8,9,10,11,3,2,12,5,4,13
90 GOTO 90
```

Para interromper sua execução basta digitar CONTROL+STOP.

A linha 10 ativa a tela gráfica de alta-resolução.O laço 20-70 lê 11 cores através da instrução READ (na linha 30). Note que a linha 80(DATA) relaciona 11 códigos que são lidos sequencialmente toda vez que o contador I é incrementado. Esta linha DATA, por sinal, pode ser colocada em qualquer ponto do programa. O laço 40-60 desenha dez linhas horizontais na cor escolhida.

Introduza uma linha OPEN no começo do programa e uma linha 35 do tipo:

```
35 PSET (?,?); PRINT # 1,A
```

que escreva o número do código de cor ao lado de cada faixa colorida. A coordenada y do PSET deve ser em função de C de maneira a imprimir o código ao lado da faixa colorida. A coordenada de x deve, obviamente, ser maior que 110 (estude a linha 50 para perceber o porquê).

7.2- Substitua as linhas 40, 50 e 60 do programa da questão anterior de maneira a obter o mesmo efeito com um único comando. (dica: pense na opção BF do LINE).

7.3- Digite o programa da figura 7.9 e rode-o.

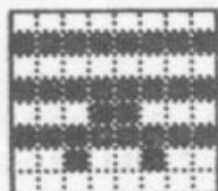
Figura 7.9

```
100 COLOR 15,1,1
110 SCREEN 2,0
120 X1=64*RND(-TIME)+100:Y1=48*RND(-TIME
)+100:X2=100:Y2=100
130 FOR I=1 TO 8
140 READ A$
150 S$=S$+CHR$(VAL("&B"+A$))
160 NEXT I
170 SPRITE$(1)=S$
180 DATA 11111111
190 DATA 11111111
200 DATA 11111111
210 DATA 11111111
220 DATA 11111111
230 DATA 11111111
240 DATA 11111111
250 DATA 11111111
260 PUT SPRITE 11,(X1,Y1),8,1
270 GOTO 270
```

Como você pode notar, o programa gera um SPRITE vermelho (cinza se seu monitor de TV não for a cores) na forma de um quadrado cheio, e coloca nas coordenadas X1,Y1, escolhida ao acaso pela linha 120.

Vamos tentar acrescentar mais linhas a este programa para transformá-lo num jogo tipo combate aéreo da Primeira Guerra Mundial. Para isso, esse SPRITE fará o papel de alvo e queremos que ele pareça um triplano (o famoso FOKKER DR-1 do Barão Von Richtofen).

Os pontos acesos e apagados do SPRITE vermelho que acabamos de definir, deverão acompanhar o esquema da figura 7.10.

Figura 7.10

Como você já sabe, um ponto aceso corresponde ao 1 e apagado ao 0. Liste o programa e levando o cursor até as linhas 180-250 altere os 1 e 0 até

conseguir a figura desejada. Lembre-se que uma alteração de uma linha na tela só é armazenada na memória se teclarmos RETURN com o cursor nesta linha.

Rode o programa alterado até se certificar que o SPRITE tem a forma da figura 7.10.

7.4- Continuando o programa iniciado no exercício anterior, precisamos agora definir um SPRITE que funcione como alça de mira do seu avião (SOPWITH CAMEL do capitão BROWN, que abateu o BARÃO VERMELHO em 1918!).

Para isso devemos acrescentar as linhas listadas na figura 7.11.

Figura 7.11

```
270 FOR I=1 TO 8
280 READ A$
290 W$=W$+CHR$(VAL("&B"+A$))
300 NEXT I
310 SPRITE$(2)=W$
320 DATA 11111111
330 DATA 11111111
340 DATA 11111111
350 DATA 11111111
360 DATA 11111111
370 DATA 11111111
380 DATA 11111111
390 DATA 11111111
400 RESTORE
410 PUT SPRITE 10,(X2,Y2),2,2
420 GOTO 420
```

Note que muitas destas linhas são iguais ou muito parecidas às linhas já digitadas no exercício 7.3.

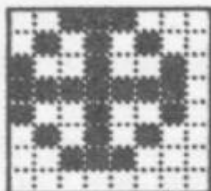
Para economizar tempo, liste o programa anterior, leve o cursor até as linhas que interessa reproduzir, altere sua numeração (e eventualmente alguns algarismos ou letras nelas contidos) e aperte RETURN. Desta forma a linha antiga permanecerá na memória e uma nova, com as alterações, será gerada.

Rode agora o programa completo e você deverá ver um SPRITE vermelho com a forma do triplano e um verde ainda com a forma de um quadrado.

Vamos agora alterar as linhas 320 a 390 de maneira a obter "uma alça de mira".

Uma sugestão de desenho está na figura 7.12.

Rode o programa assim alterado para ver se os dois SPRITES têm a forma desejada.

Figura 7.12

7.5- Vamos agora digitar o resto do programa de maneira a movimentar o triplano ao acaso (estratégia de fuga) e a alça de mira orientada pelas setas.

Acrescente ao que já foi digitado as linhas listadas na figura 7.13.

Figura 7.13

```
420 SPRITE ON
430 IF STRIG(0) THEN N=N-1:PLAY"S8M80o1C
4"
440 X1=X1+3*INT(RND(1)+.5)-3*INT(RND(1)+
.5)
450 Y1=Y1+3*INT(RND(1)+.5)-3*INT(RND(1)+
.5)
460 IF X1<0 OR X1>255 OR Y1<0 OR Y1>192
THEN X1=128:Y1=96
470 A=STICK(0)
480 Y2=Y2+(A=8)+(A=1)+(A=2)-(A=6)-(A=5)-
(A=4)
490 X2=X2+(A=8)+(A=7)+(A=6)-(A=2)-(A=3)-
(A=4)
500 PUT SPRITE 10,(X2,Y2),2,2
510 PUT SPRITE 11,(X1,Y1),8,1
520 ON SPRITE GOSUB 570
530 B=B+1
540 IF B>200 THEN 600
550 GOTO 420
560 END
570 IF STRIG(0) THEN PLAY"V15o5C16":N=N+
4:PSET (X1+4,Y1+4)
580 SPRITE OFF
590 RETURN
600 OPEN "GRP:" AS#1
610 PSET (10,10),1
```

```
620 PRINT #1,"VOCE FEZ";N;"PONTOS!"
630 FOR I=1 TO 3000:NEXT I
640 RUN
```

Digite todas as linhas com muito cuidado (por exemplo: não confunda 0 por O!) e rode o programa para ver o que ele faz.

O objetivo é colocar a alça de mira sobre o triplano (movimentando as setas). Neste momento você deve apertar a barra de espaços para atirar. Toda vez que você atirar, ouvirá um som grave. Se você acertar o alvo ouvirá também um som agudo e um ponto branco ficará marcado na tela.

Decorrido um certo tempo aparecerá a mensagem relatando quantos pontos você fez.

A tarefa deste exercício, além da digitação, é bem simples e agradável:jogue um pouco e divirta-se!

7.6- Agora chega de brincadeiras! Vamos entender como o jogo funciona.

Na linha 430 o computador verifica se a barra de espaço (STRIG (0)) foi pressionada. Em caso afirmativo é descontado um ponto (N=N-1) e toca o som grave (PLAY ...).

a) Se você possuir um joystick, altere a função STRIG(0) para STRIG(1) e responda como o tiro é dado com essa alteração.
b) Que alteração você faria nesta linha para que cada tiro implique na perda de 3 pontos ao invés de 1?

7.7- Nas linha 440 e 450 as coordenadas X1 e Y1 podem sofrer, ao acaso, uma alteração para mais ou para menos de 3 unidades. Lembre-se que:

INT(RND(1)+.5)

pode valer, ao acaso, ou 0 ou 1.

Isso faz com que o triplano tenha o movimento "browniano" que determina sua estratégia de fuga.

Cuidado, o termo "browniano" se refere ao botânico escocês Robert Brown (1773-1858) que descobriu o movimento desordenado das moléculas nos líquidos e não tem nada a ver com o nosso herói, capitão BROWN da RAF canadense.

a) Digamos que você queira aumentar a dificuldade do jogo, produzindo um movimento desordenado do triplano com o dobro da velocidade. Que alteração você deveria introduzir nas linhas 440 e 450 ?

b) Digamos agora que você queira produzir uma versão deste jogo para uma criança pequena, reduzindo a velocidade do avião a um terço da original. Que alteração você introduzirá nas linhas 440 e 450 ?

7.8- A linha 460 checa se o avião, por acaso, não saiu da tela. Se isso ocorrer, ela o recoloca no centro.

Que alteração você faria para que ele se movimentasse apenas no quarto superior esquerdo da tela e, ao sair dele, fosse recolocado no meio desta região? (Veja figura 7.14)

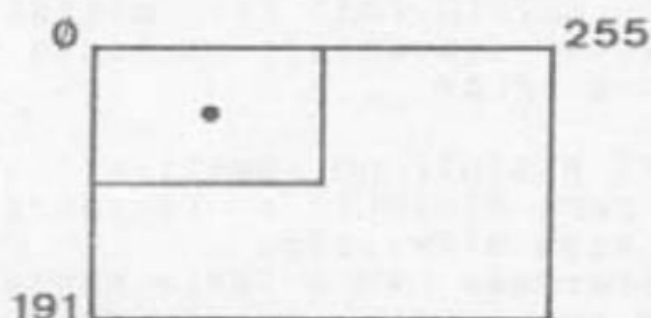
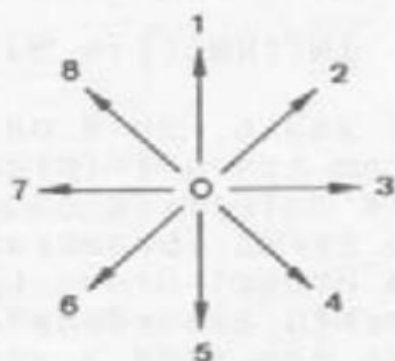
Figura 7.14

7.9- A linha 470 lê as setas do teclado atribuindo à variável A um valor entre 1 e 8 em função das setas pressionadas. (Veja a figura 7.15)

Figura 7.15

As diagonais são obtidas ao se pressionar uma seta vertical e uma horizontal simultaneamente.

A expressão entre parênteses, por exemplo, (A=5) valerá 1 se a seta vertical correspondente estiver sendo pressionada e 0 em caso contrário.

As linhas 480 e 490 alteram o valor das coordenadas X2 e Y2 da alça de mira em função da setas pressionadas.

Que alteração você faria neste trecho do programa para que, ao invés das setas, você pudesse movimentar a alça de mira com o joystick 1? (Consulte a função STICK no manual do micro).

7.10-As linhas 500 e 510 colocam o avião e a alça na tela em função das novas coordenadas, atualizadas ao acaso para o triplano e pelas setas no caso da mira.

A linha 530 incrementa uma variável contadora (B) e a linha 540 checa se ela ultrapassou o valor 200. Em caso afirmativo ela desvia a execução do programa para uma sub-rotina finalizadora.

Que alteração você faria na linha 540 para reduzir o tempo de jogo à metade?

7.11-A linha 420 ativa, no micro, um sistema de alarme que só funcionará se dois SPRITES se "chocarem" na tela.

A linha 520 põe em funcionamento este alarme toda vez que você consegue colocar a alça de mira em cima do triplano.

Neste caso (mira sobre o triplano), ela desvia a execução do programa para a sub-rotina que começa na linha 570.

```
570 IF STRIG(0) THEN PLAY"V15o5C16":N=N+
4:PSET (X1+4,Y1+4)
580 SPRITE OFF
590 RETURN
```

Na linha 570 o micro checa se a barra de espaços está sendo pressionada (ou seja, se o capitão BROWN está atirando!). Neste caso ela toca um aviso sonoro mais agudo, aumenta em 4 pontos a contagem do jogador e marca um ponto branco na tela (nuvenzinha de explosão, se você tiver imaginação!) assinalando o local do impacto.

A seguir o alarme de "choque" de SPRITES é

desativado (se não você ficaria dando tiros indefinidamente em cima de um pobre barão "congelado") e o RETURN ratoma a execução do programa principal.

a) O que você alteraria na linha 570 para poder jogar com o joystick 1?
b) Que alteração fazer para que cada tiro certeiro valha 10 pontos?

7.12-Vamos agora analisar a sub-rotina finalizadora:

```
600 OPEN "GRP:" AS #1
610 PSET (10,10),1
620 PRINT #1,"VOCE FEZ";N;"PONTOS!"
630 FOR I=1 TO 3000:NEXT I
640 RUN
```

A linha 600 abre um arquivo na tela de alta-resolução (GRP:) como sendo o de número 1 (#1).

A linha 610 localiza o cursor para que a linha 620 escreva a mensagem a partir das coordenadas (10,10).

A linha 630 é um "laço vazio" que simplesmente faz a mensagem ficar congelada na tela por um certo tempo antes que a linha 640 reinicie o jogo.

a) Altere a linha 610 de maneira a fazer a mensagem surgir no meio da tela.
b) Introduza uma linha 625 que cheque se o jogador fez mais que, digamos, 100 pontos. Em caso afirmativo deverá aparecer a mensagem "PARABÉNS!" em baixo da mensagem original.
c) Altere a linha 630 de maneira a "congelar" a mensagem pela metade do tempo.

7.13-Experimente alterar a linha 110 para:

```
110 SCREEN 2,1
```

e rode o programa.

a) O que aconteceu?
b) Que alteração fazer no final da linha 570 para que as "nuvenzinhas" surjam no meio da nova alça de mira?

AULA 8

Assunto : Recursos Sonoros.

Objetivo : Aprender a usar a função PLAY.

SONS

O MSX possui um Gerador Programável de Sons (PSG) que permite facilmente, através do comando PLAY, produzir sons e músicas.

Os sons no comando PLAY podem ser produzidos em três canais independentes e são especificados por subcomandos representados por "strings" dentro de "" ou por variáveis "string".

A sintaxe do comando PLAY é :

```
PLAY string 1, string 2, string 3
```

onde, string 1 = subcomandos do canal 1.
string 2 = subcomandos do canal 2.
string 3 = subcomandos do canal 3.

Os subcomandos utilizados pelo comando PLAY estão relacionados na figura 8.1. Para maiores detalhes sobre a notação musical, aconselhamos a leitura do "CURSO DE MÚSICA EM MSX" desta mesma Editora.

As notas musicais são indicadas de acordo com a notação cifrada, ou seja, a letra A corresponde à nota musical Lá, B à nota Si, C ao Dó, E ao Ré, F ao Mi e a letra G ao Fá. As notas e os semi-tons são especificados na figura 8.2.

A oitava é selecionada por On, onde n pode variar de 1 a 8. Cuidado par não confundir o O de Oitava com o algarismo 0! Na ausência desta indicação, o computador assume n=4 (quarta oitava). A figura 8.3 mostra a correspondência entre as oitavas e o teclado de um piano.

Figura 8.1 - Subcomandos da função PLAY.

SUBCOMANDO	VALORES PERMITIDOS	FINALIDADE
Tn	n = 32 a 255	Determinar o tempo (andamento) da música.
On	n = 1 a 8	Determinar uma das 8 oitavas do MSX BASIC.
Ln	n = 1a 64	Determinar a duração da nota.
Nn	n = 0 a 96	Especificar uma nota musical através de sua numeração absoluta.
A.B.C.D.E.F.G		Indicar a nota musical dentro de uma oitava pré-determinada.
# ou +		Sustenido
—		Bemol
Rn	n = 1 a 64	Determinar uma pausa.
(ponto)		Aumentar a duração de uma nota ou pausa em 50%.
Vn	n = 0 a 15	Determinar o volume.
Mn	n = 0 a 65535	Determinar o período da variação do volume durante a execução de uma nota.
Sn	n = 0 a 15	Determinar a forma de variação do volume durante a execução de uma nota.

Figura 8.2 - Escala Musical.

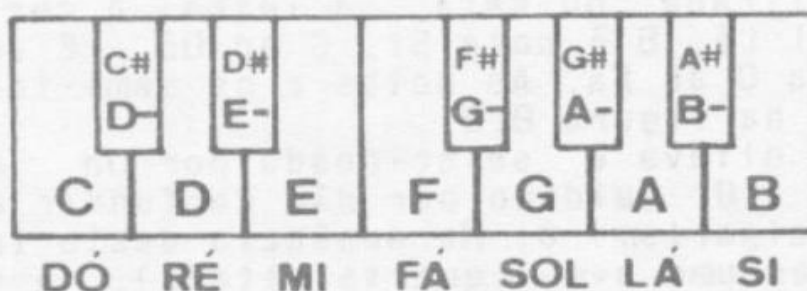



Figura 8.3 - Correspondência entre as oitavas e o teclado de um piano.

A notação Nn permite usar uma numeração absoluta referente ao teclado musical completo. O número n deve estar compreendido entre 0 e 96. A correspondência entre os números das notas e as teclas de um piano é mostrada na figura 8.4.

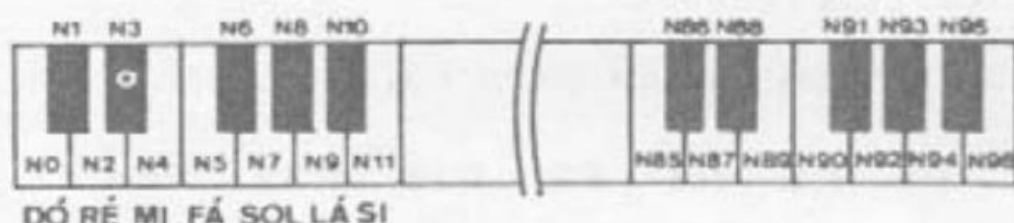
Figura 8.4 - Correspondência entre os números e as teclas de um piano.

A notação Ln especifica a duração de uma nonota. onde n é um número inteiro compreendido entre 1 e 64 e o seu valor tem o significado apresentado na Tabela 8.5.

Figura 8.5 - Duração de uma nota.

L1	n = 1 : indica uma nota semibreve (nota de maior duração)
L2	n = 2 : indica uma nota mínima (2 mínimas = 1 semibreve)
L4	n = 4 : indica uma nota semimínima (4 semimínimas = 1 semibreve)
L8	n = 8 : indica uma nota colcheia (8 colcheias = 1 semibreve)
L16	n = 16 : indica uma nota semicolcheia (16 semicolcheias = 1 semibreve)
L32	n = 32 : indica uma nota fusa (32 fusas = 1 semibreve)
L64	n = 64 : indica uma nota semifusa (64 semifusas = 1 semibreve)

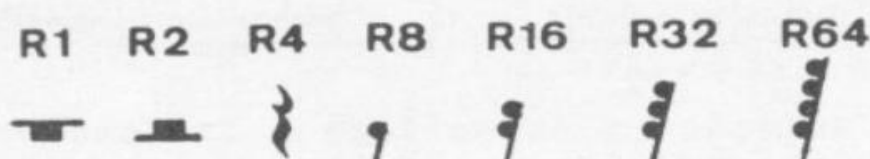
Assim como nas partituras musicais, para indicar uma duração de uma vez e meia a duração original da nota, basta escrever a letra que indica a nota seguida de um ponto (.).

Se acrescentarmos mais de que um ponto, a cada um corresponderá um aumento na duração da nota de 50%. Por exemplo, no caso de 3 pontos sucessivos, a duração final da nota é igual à original multiplicada por 3.375 (3.375=1.5*1.5*1.5).

Para indicarmos uma pausa musical utilizamos Rn, onde n é um número inteiro compreendido entre 1 e 64. O critério para a duração de uma pausa é o mesmo que o da duração de uma nota. Isto é, R1 indica uma pausa equivalente à duração de uma semibreve (L1), R2 indica uma pausa que equivale à duração de uma breve (L2), e assim por diante até R64.

A figura 8.6 mostra as pausas e seus símbolos símbolos correspondentes.

Figura 8.6 - Pausas e seus símbolos correspondentes.

R1 R2 R4 R8 R16 R32 R64

Como no caso das notas, o ponto aumenta a duração da pausa em 50%.

O tempo (ou andamento) da música é especificado por Tn, onde n pode variar de 32 a 255. Quanto maior for n, mais rapidamente será executada a música. Na ausência de indicação de tempo, o valor assumido é 120.

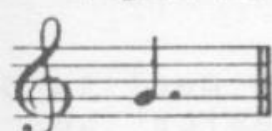
O volume do som é dado por Vn, onde n pode valer algo entre 0 e 15, sendo o seu valor inicial igual a 8.

É possível omitir a letra L e escrever a duração da nota (n) após o seu nome. Assim,

PLAY "L4O5A" equivale a PLAY "O5A4"

A duração (L) e a oitava (O), se forem omitidas, assumem o último valor especificado.

Depois do exposto até aqui, temos condições de transcrever notas de um pentagrama para a linguagem musical do BASIC-MSX. Para exemplificar, veja a figura 8.7.

Figura 8.7 - Transcrição de notas musicais de um pentagrama.

PLAY"L4O4G." equivale a PLAY"L4N43."

PLAY"L8O4D+" equivale a PLAY"L8N39"

Para finalizar e exemplificar tudo o que foi dito sobre o comando PLAY, execute o programa da figura 8.8.

Figura 8.8

```
100 REM TURKEY IN THE STRAW
110 PLAY"T250S0M5000","T250S0M5000","T25
0S0M5000"
120 PLAY"o4E8D8","R4","R4"
130 PLAY"C4L8CDCo3GEF","o4L4CGEG","o5R4L
4CR4C"
140 PLAY"GAGEG4o4C8D8","CGEG","R4CR4C"
150 PLAY"E4E4L8EDCD","CGEG","R4CR4C"
160 PLAY"E4D4R4E8D8","o3Go4Do3Bo4D","R4o
4GR4G"
170 PLAY"C4L8CDCo3GEF","CGEG","R4o5CR4C"
180 PLAY"GAGEG4o4C8D8","CGEG","R4CR4C"
190 PLAY"E8G4L8AGECD","CGEG","R4CR4C"
200 PLAY"L4EDCR4","Co3Bo4CR4","o4GGGR4"
210 PLAY"E8G4E8G4G4","CGEG","R4o5CR4C"
220 PLAY"E8G4E8G4G4","CGEG","R4CR4C"
230 PLAY"F8A4F8A4A4","o3Go4Co3Ao4C","R4o
4FR4F"
240 PLAY"F8A4F8A4o3A8B8","o3Go4Co3Ao4C",
"R4FR4F"
250 PLAY"o4L4CCo3GG","CGEG","R4o5CR4C"
260 PLAY"EEDL8CD","D#R4o3BR4","o4A#R4GR4
"
270 PLAY"EG4L8AGECD","o4CR4CR4","GR4GR4"
280 PLAY"L4EDCR1","Co3Bo4CR4","GGGR4"
```

Experimente agora, fazer a alteração da figura 8.9.

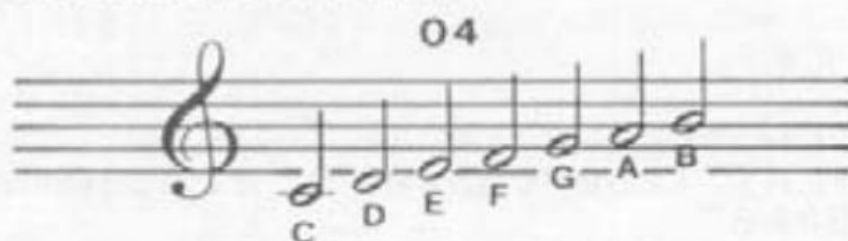
Figura 8.9

```
110 PLAY"T250S13M700","T250S13M700"," T25
0S13M700"
```

EXERCÍCIOS

8.1- Muitas vezes você deve ter visto uma partitura musical e teve vontade de "ouvir" a música correspondente àquele monte de "bolinhas" desenhadas no pentagrama.

Vamos aprender agora como ler a partitura na clave de Sol.

A oitava 4 (O4) que é o valor que o computador assume ao ser ligado, é representado no pentagrama da seguinte forma:

Vamos tentar transformar um pedaço da partitura do famoso PIRULITO num programa de computador.

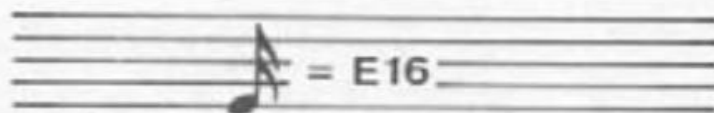
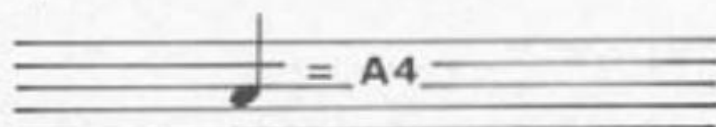
Inicialmente vamos digitar uma linha que dê ao computador o som de piano: (cuidado para não confundir Ø com O).

```
10 PLAY "SØM5ØØØO4"
```

Na linha seguinte:

```
20 PLAY "..........
```

Coloque a sucessão de notas, cada uma seguida do número que dá sua duração.

Apenas para exemplificar, a primeira nota será:

enquanto que a última deverá ser:

Após completar a linha 20, não esqueça de fechar as aspas.

Ao comandar RUN, você deverá ouvir as primeiras notas do "Pirulito que bate, bate..."!

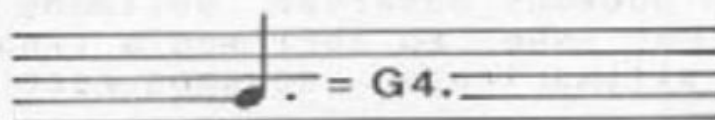
8.2- Vamos continuar a música do exercício anterior. A sucessão de notas correspondente à segunda frase: "... pirulito que já bateu." é dada por:

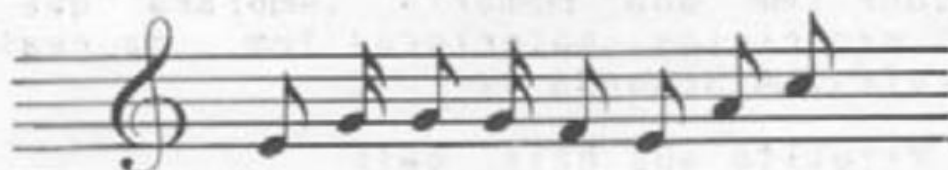
Comece então numa linha:

```
30 PLAY "......
```

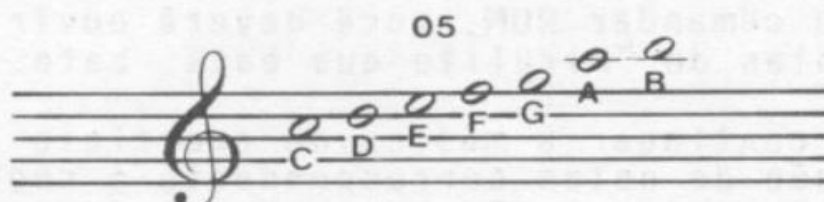
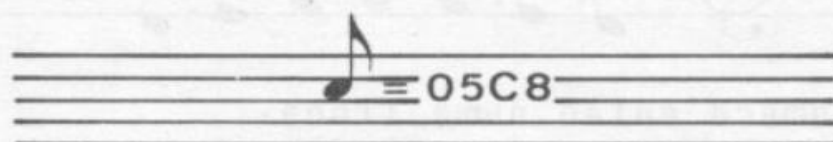
e coloque nela a sucessão de notas correspondente à segunda frase.

Como a última nota tem um ponto depois, isso significa que ela deve durar uma vez e meia o valor indicado.

Para isso basta colocar um ponto depois do seu código:

Mais uma vez feche a linha com aspas, digite RETURN e a seguir RUN. Ouça agora a sua musiquinha um pouco mais completa.

8.3- Agora vamos continuar a musiquinha com o resto de sua partitura. O trecho "Quem gosta de mim é ela ..." corresponde a esta sequência de notas:

Abra a linha

```
40 PLAY "........
```

com esta sequência. Cuidado, porém, com a última nota, ela escapa da oitava 4 e entra na oitava 5:

Por isso, quando for digitar o código da última nota, você deve digitar antes O5, indicando ao computador que estamos subindo uma oitava:

8.4- Está faltando o "quem gosta dela sou eu!"
A sequência das notas é

Como podemos observar, voltamos para a oitava 4 (O4). Por isso, ao abrirmos a linha 50 para digitar a última frase, devemos recolocar o micro em O4:

```
50 PLAY "O4...
```

Mais uma vez cuidado para não confundir o O com 0!
Digitando RUN, você tem agora a música completa!

8.5- Se você tem boa memória, lembrará que a música dos exercícios anteriores tem, na realidade, 2 estrofes. A segunda é:

Pirulito que bate, bate
Pirulito que já bateu.

A menina que eu amava
Coitadinha já morreu.

Se você quiser omitir este final triste, pode gravar seu programinha em fita do jeito que está.

Se quiser cantarolar a música completa, você deverá repetir as linhas 20, 30, 40 e 50. Para não ter que digitar tudo de novo nas linhas 60, 70, 80 e 90, temos duas soluções: ou acrescentamos um laço:

```
15 FOR N=1 TO 2
55 NEXT N
```

que fará o micro repetir as 4 frases duas vezes ou então criamos as linhas 60, 70 80 e 90 usando os recursos de edição de tela. Para isso basta limpar a tela (SHIFT + HOME/CLS), listar o programa com LIST (e RETURN), levar o cursor até o número 20 da linha 20, mudá-lo para 60 e pressionar RETURN. A seguir mudar com o mesmo processo o número 30 para 70, o 40 para 80 e o 50 para 90. Não esqueça do RETURN após cada alteração. Listando novamente o programa o vemos completo da linha 10 à 90.

Tente as duas soluções e, após cada uma delas comande:

```
? FRE(0)
```

que fornece o número de bytes livres na RAM. Anote esse número nos dois casos e responda, qual das duas soluções consome menos memória do micro?

8.6- Como seu micro tem 3 vozes, podemos transcrever partituras mais complexas e obter efeitos mais bonitos que o da musiquinha discutida nos exercícios anteriores.

Vamos pegar agora uma música barroca do século XVIII, o famoso "GREENLEAVES" com uma partitura em 3 vozes. A única dificuldade adicional na transcrição desta partitura é o sinal # (sustenido).

Quando aparecer um sinal desse ao lado de uma nota, basta digitar um sinal + após seu código. Se, em outra partitura, aparecer o sinal b (bemol), digite - após o código.

Figura 8.10 - Partitura da música "Greenleaves"

Vamos digitar esta música usando uma linha de BASIC para cada compasso. A título de exemplo, vamos já listar os 5 primeiros compassos (linhas de 10 a 80). Na linha 5, colocamos os dados que determinam o volume e andamento da música.

```
5 PLAY"V15T180","V12T180","V10T180"
10 PLAY"o4A4","o4R4","o4R4"
20 PLAY"o5C2D4","A2.","E2."
30 PLAY"E4.F8E4","o5C2.","A2."
40 PLAY"D2o4B4","o4G2.","B2G4"
50 PLAY"G4.A8B4","B4.A8G4","D2."
60 PLAY"Aqui você continua...
```

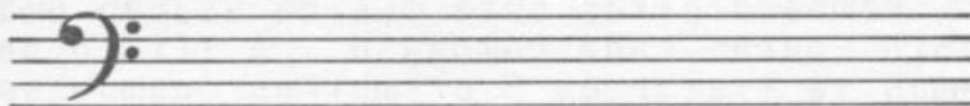
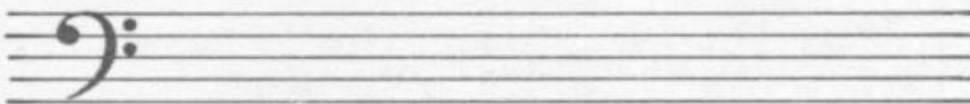
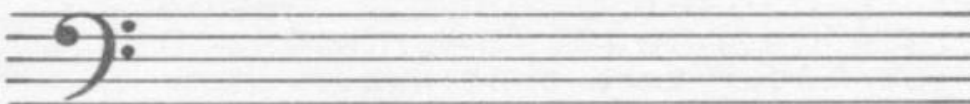
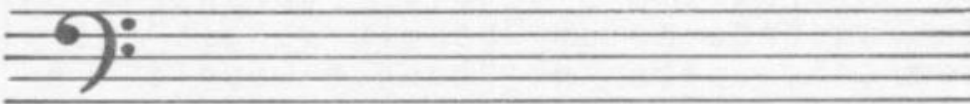
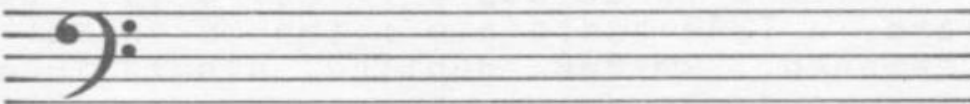
8.7- Para finalizar, vamos inverter o tipo de exercício: o programa a seguir reproduz a partitura da mão esquerda de um "boogie-woogie" no piano (para quem não sabe o "boogie-woogie" é o pai do "rock'n'roll", portanto o avô do rock que se escuta atualmente!).

A linha 10 coloca a primeira voz com som de piano: o T240 acelera o ritmo (o normal é T120) e o comando S8M1000 "duplica" a nota tocada, dando o sincopado do acompanhamento.

Como você pode notar as linhas 20, 30 e 50 são idênticas, bastando digitar a linha 20, reproduzindo-a depois com edição de tela.

```
10 PLAY"T240S8M1000"
20 PLAY"o2CEGAA+AGE"
30 PLAY"o2CEGAA+AGE"
40 PLAY"FAo3CDD+DCo2A"
50 PLAY"o2CEGAA+AGE"
60 PLAY"o2DEFF+GFED"
70 GOTO 20
```

Digite o programa, ouça o que ele faz, brinque um pouco com o S8M1000 alterando esses valores para obter outros timbres. Agora vamos à tarefa séria: escreva a partitura correspondente às 5 linhas de BASIC. Para isso vamos aprender outra clave: a de FÁ.

Tente agora fazer sua transcrição:

LINHA 20
LINHA 30
LINHA 40
LINHA 50
LINHA 60
L1 L2 L4 L8 L16 L32 L64

APÊNDICE A – Os Teclados Escondidos

Nas tabelas a seguir (uma para o EXPERT e outra para o HOTBIT) você tem todos os 256 caracteres relacionados com seus códigos hexadecimais.

Por exemplo, se você quiser saber qual o caractere de código hexadecimal 9C, basta procurar na linha 9, coluna C, e você encontrará o símbolo da libra esterlina. Se você comandar

```
PRINT CHR$(&H9C)
```

obterá esse símbolo na tela.

No campo abaixo do caractere, você encontra a combinação de teclas que devem ser pressionadas, simultaneamente, para obtê-lo via teclado (figura (a)).

No caso da libra esterlina, por exemplo, na tabela do EXPERT você encontra o correspondente à figura (b). Isto significa que, para obter o símbolo da libra esterlina você deverá pressionar, simultaneamente, as teclas 4+RGRA+SHIFT. Já na tabela do HOTBIT, você encontra o apresentado na figura (c). Neste caso, para obter o símbolo da libra, você deverá pressionar simultaneamente as teclas R+CODE.

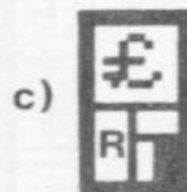
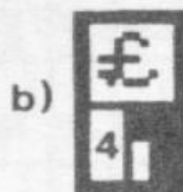
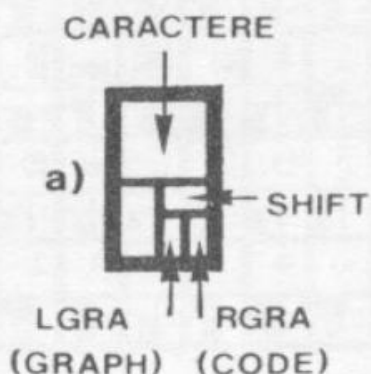
Em alguns casos, você encontra o campo da tecla totalmente preenchido (como por exemplo, no caractere 7F). Isto significa que esse caractere não pode ser obtido através do teclado.

De qualquer forma você pode imprimir um caractere na tela usando o comando

```
PRINT CHR$(&H linha coluna)
```

Esta regra só tem uma exceção: para imprimir na tela os caracteres semi-gráficos das duas primeiras linhas (ou seja, de 00 a 1F), você deve comandar

```
PRINT CHR$(1)+CHR$(64+&H linha coluna)
```

Com relação aos caracteres acentuados, você pode obtê-los também digitando o acento e a seguir a letra correspondente. No HOTBIT, esta é a única maneira de conseguí-los, no caso de na tabela aparecer o símbolo ★ .

TECLADO DO EXPERT (GRADIENTE)

TECLADO DO HOTBIT (SHARP)

APÊNDICE B – O Uso do Gravador

Quando você desliga o computador tudo o que estiver armazenado na memória RAM será apagado.

Assim, se você quiser preservar o seus pro-programas ou arquivos gerados por outros programas, deve gravá-los em fita cassete ou disco magnético antes de desligar o microcomputador. Dessa maneira será possível recarregá-los na memória posteriormente.

O sistema de armazenamento em fita cassete não é o mais perfeito possível pois apresenta muitas falhas: falta de confiabilidade, demora no tempo de gravação e leitura e acesso sequencial. Por outro lado, ele se constitui no sistema mais econômico de arquivo.

Sendo um sistema sujeito a falhas, é bastante importante que você tome alguns cuidados para minimizá-las:

1) Utilize um bom gravador.

2) Utilize fitas de boa qualidade, que reproduzam bem os agudos, não soltem emulsão magnética e que não sejam de longa duração.

3) Mantenha a cabeça magnética e o rolo pressor do gravador sempre limpos. Utilize álcool isopropílico para a limpeza e espere secar para introduzir a fita.

4) Desmagnetize periodicamente o cabeçote. Se você não possuir um desmagnetizador, leve o gravador em uma oficina técnica.

5) Guarde suas fitas em locais secos, limpos e afastados de fontes de interferência magnética tais como televisores, caixas acústicas, transformadores, reatores de lâmpadas fluorescentes, ímãs, etc.

6) Rebobine sempre suas fitas após o uso.

7) Faça várias cópias de seus principais programas em fitas diferentes.

COMANDOS DE ARMAZENAMENTO E LEITURA

Para gravar e carregar programas em MSX BASIC e em Linguagem de Máquina devem-se utilizar comandos específicos descritos a seguir.

a) CSAVE e CLOAD

Para gravar em fita cassete um programa escrito em BASIC, utilize o comando CSAVE. Sua sintaxe é:

CSAVE "nome"

Onde, "nome" é um conjunto de no máximo 6 caracteres e deve estar sempre entre aspas (").
O procedimento de gravação é o seguinte:

* Digite CSAVE "nome" mas ainda não pressione a tecla RETURN.

* Acione as teclas PLAY e RECORD de seu gravador com a fita devidamente posicionada.

* Pressione a tecla RETURN.

A gravação estará completa quando aparecerem a palavra "Ok" e o cursor na tela.
Nesse ponto, você pode testar se a sua gravação foi bem feita utilizando o comando CLOAD?. Para isso, rebobine a fita, comande CLOAD? e acione a tecla PLAY do gravador.
Não se esqueça de ajustar o volume de seu gravador. Um nível médio, a princípio, é adequado.
Ao executar CLOAD?, deve aparecer na tela, logo no início da reprodução do trecho da fita que

contem o programa gravado, a mensagem:

Achei: nome dado ao programa (HOTBIT)
Found: nome dado ao programa (EXPERT)

Se isso acontecer, tudo está correndo bem. Espere então até o final da reprodução. Deverão surgir novamente a palavra "Ok" e o cursor, indicando que a gravação está perfeita.

Se algo der errado, tente novamente com um ajuste de volume em seu gravador ligeiramente diferente do anterior.

Se após algumas tentativas, você não obtiver sucesso, é provável que a gravação não esteja boa e, nesse caso, é interessante efetuar novamente a gravação do programa armazenado na memória do micro.

Quando você quiser carregar um programa gravado com CSAVE na memória do seu MSX, utilize o comando CLOAD seguindo o mesmo procedimento descrito anteriormente para o comando CLOAD?. A diferença entre os dois é que o primeiro carrega um programa na memória do micro apagando o anterior e o segundo apenas verifica se o que está na memória é idêntico ao que está gravado na fita.

O comando CLOAD pode ser utilizado de duas formas:

CLOAD

ou

CLOAD "nome"

Na primeira forma, será lido o primeiro programa encontrado na fita. Na segunda, será lido apenas o programa especificado pelo nome dado. Caso exista outro programa com um nome diferente gravado na fita antes do desejado, deve surgir a mensagem:

Pulei: outro nome (HOTBIT)
Skip: outro nome (EXPERT)

Isso acontece até que seja achado o programa especificado por "nome", quando finalmente surge a mensagem:

Achei: nome (HOTBIT)
Found: nome (EXPERT)

Espere o fim da leitura, ou seja, aguarde a

liberação do cursor e a palavra "Ok".

Novamente problemas de ajuste de volume do gravador podem ocorrer. Se você não obtiver sucesso numa primeira tentativa, não desista. Tente novamente com outros níveis de ajuste.

Se mesmo assim você não conseguir carregar seus programas, talvez seja necessário fazer um ajuste do alinhamento do cabeçote de seu gravador (azimute). Para isso, localize o furo que dá acesso ao parafuso de ajuste (ao lado da cabeça de gravação) e, desconectando todos os cabos, ponha uma fita com o programa que você que carregar. Reproduza essa fita e gire o parafuso de ajuste, com o auxílio de uma chave de fenda, no sentido horário ou anti-horário até obter o som mais estridente possível.

b) SAVE e LOAD

Os comandos SAVE e LOAD tem quase a mesma finalidade que CSAVE e CLOAD. A diferença está na sintaxe e na possibilidade de introduzir-se um parâmetro que permite a execução automática do programa assim que for carregado. Além disso, a gravação é feita no formato ASCII, o que possibilita o uso do comando MERGE.

Para gravar, use :

```
SAVE "CAS: nome"
```

Para carregar, utilize:

```
LOAD "CAS:" ou LOAD "CAS:nome"
```

ou ainda,

```
LOAD "CAS:",R ou LOAD"CAS:nome",R
```

caso se queira a execução automática do programa quanquando ele estiver carregado.

c) BSAVE e BLOAD

Os comandos BSAVE e BLOAD são utilizados para gravar e carregar programas em Linguagem de Máquina (uma sequência de bytes).

Para utilizar-se BSAVE, deve-se informar ao micro o endereço de memória no qual o programa em Linguagem de Máquina está armazenado. Tanto o endereço

inicial quanto o final, para que o MSX BASIC saiba exatamente qual o bloco de memória deve ser copiado na fita.

Além disso, ao usar-se esse comando, pode-se informar também o endereço em que deve começar a execução do programa, caso esse não seja igual ao endereço inicial.

Sua sintaxe é:

BSAVE "CAS:nome", endereço inicial, endereço final, endereço de início da execução

Para carregar um programa em Linguagem de Máquina, use o comando BLOAD cuja sintaxe é:

BLOAD "CAS:" ou BLOAD "CAS:nome"

Se você desejar a execução automática do programa após o carregamento, basta acrescentar a letra R, ou seja:

BLOAD "CAS:",R ou BSAVE "CAS:nome",R

APÊNDICE C — RESPOSTAS

1.2 - c   1.3 - a   1.4 - d   1.5 - f   1.6 - e

2.1 -

Código	33	34	75	99	129
Caractere	!	"	K	c	ü

2.2 - Cada caractere é mostrado na tela durante o dobro do tempo em relação ao do programa original.

2.3 - a) Os caracteres de código menor que 33 também são mostrados.

b) ■

2.4 - O programa volta ao início de sua execução e fica "rodando" indefinidamente.

2.5 - c

3.1 -
```
ESTE
TEXTO
SERVE
COMO
TESTE
```

3.2 - A vírgula divide a tela em duas partes. A primeira começa na coluna 0 e termina na coluna 13. A segunda vai da coluna 14 até o fim.

Cada vírgula colocada após o comando PRINT desloca a próxima impressão para a parte seguinte que esteja livre.

O resultado que você deve obter é:
```
0123456789012345678901234567890123456789

ESTE          TEXTO
SERVE         COMO
TESTE
Ok
■
```

Experimente acrescentar mais duas vírgulas na linha 20. Você obterá:
```
0123456789012345678901234567890123456789

ESTE
              TEXTO
SERVE         COMO
TESTE
Ok
■
```

3.3 - ESTETEXTOSERVECOMOTESTE

```
3.4 -  012345678901234567890
       0
       1 ESTE
       2
       3   TEXTO
       4
       5     SERVE
       6
       7       COMO
       8
       9         TESTE
       0
```

```
3.5 - 10 CLS                 40 PRINT"SERVE";
      15 LOCATE 1,1          45 LOCATE 6,4
      20 PRINT"ESTE";        50 PRINT"COMO";
      25 LOCATE 5,2          55 LOCATE 1,5
      30 PRINT"TEXTO";       60 PRINT"TESTE";
      35 LOCATE 10,3
```

4.1 - (1) C (2) G (3) D (4) D (5) E
(6) A (7) A (8) F (9) C (10) B

4.2 - (1) B (2) A (3) A (4) B (5) B

4.3 - 1) A constante 312 não deveria aparecer entre aspas (") para ser numérica.
2) A variável inteira (I%) não assume valores superiores a 32767.
3) A variável alfanumérica (B$) não pode ser igualada a uma constante numérica.
4) A variável numérica (AZ) não pode ser igualada a uma constante alfanumérica ("MSX") a não ser que se tenha previamente utilizado o comando DEFSTR A, como veremos no volume 2 desta coleção.
5) A variável de precisão simples (AZ!) assume o valor a ela atribuído mas não em precisão dupla (3.1D21).
6) O nome da variável corresponde a uma palavra reservada do MSX-BASIC (OR).

4.4 - 20 ?A:B=3

4.5 - As linhas 20, 30 e 40 atribuem às variáveis R$, C$ e A$ os conteúdos "RAIO=", "CIRCUNFERÊNCIA="

e "ÁREA=", respectivamente.

A linha 50 permite, através do comando INPUT, que seja atribuído o valor do raio à variável R.

A linha 90 apresenta na tela as cadeias de caracteres RAIO=, CIRCUNFERÊNCIA=, ÁREA= e os respectivos valores.

A linha 100 "imprime" uma linha em branco e desvia o programa para a linha 50 para que um novo cálculo possa ser efetuado.

4.6 - a) No mês 1032 pois ocorreu "Overflow", isto é, foi ultrapassada a capacidade de cálculos do computador.
b) Os cálculos são feitos com números inteiros e o "Overflow" ocorre após o mês 21, se forem feitas as mesmas entradas do item a.
c) Os cálculos são feitos com números de precisão simples.

5.1 - As linhas 20, 80 e 90 devem ser alteradas para:

```
20 FOR I=1 TO 6
80 PRINT "A MEDIA ARITMETICA DESTES";6
90 PRINT "NUMEROS VALE";T/6
```

5.2 - As linhas 20, 80 e 90 devem ser alteradas para:

```
20 FOR I=1 TO N
80 PRINT "A MEDIA ARITMETICA DESTES";N
90 PRINT "NUMEROS VALE";T/N
```

5.3 - Não

5.4 - A linha 40 de eria ser alterada para:

```
40 PRINT INT(9*RND(-TIME))+1
```

5.5 - Ela compara o valor do produto dos dois números que está armazenado na variável P com o valor dado como resposta via teclado contido na variável C. Se esses valores forem iguais é impressa a mensagem "MUITO BEM", caso contrário (ELSE) aparece a mensagem "ERRADO".

5.6 - A linha 110 deveria ser mudada para:

```
110 FOR I=1 TO 4
```

5.7 - No caso de dar-se uma resposta incorreta, não é impressa a mensagem "ERRADO". No lugar disto o programa desvia para a linha 100 e é reiniciado.

A principal diferença é que, com essa mudança, o usuário do programa de erá acertar todos os testes propostos para chegar ao fim. Um erro basta para que se comece tudo de novo!

A última versão do programa é inadequada para ser aplicada a uma criança pois ·lém de ser muito hostil (obriga a criança a acertar tudo), não a informa que errou.

A primeira versão é melhor, porém poderia ser

aprimorada se modificássemos a linha 210 para:

```
210 IF C=P THEN PRINT"MUITO BEM" ELSE PR
INT"VOCE ERROU, TENTE DE NOVO":GOTO 190
```

Dessa forma, se for cometido um erro, o programa pede para que a nova tentativa seja feita.

Mas ainda há um problema: o programa só propõe a próxima conta se a criança acertar, não importando quantos erros ela cometa para isso.

Vamos limitar o número de erros numa certa conta em três, por exemplo. Para isso modifique a linha 210 para:

```
210 IF C=P THEN PRINT"MUITO BEM" ELSE GO
TO 240
```

Acrescente também:

```
115 E=0
235 END
240 E=E+1
250 IF E<=3 THEN PRINT "ERRADO, TENTE DE
NOVO":GOTO 190
260 PRINT "ERRADO, A RESPOSTA é";P:GOTO
220
```

Se agora for cometido um erro, o programa desvia para a linha 240 que incrementa em uma unidade a variável E (contadora do número de erros). A linha 250 testa se o número de erros cometidos é menor ou igual a 3. Se for, pede-se que seja feita uma nova tentativa e desvia-se o programa para a linha 190. Caso contrário, não ocorre desvio e é executada a linha 260 que imprime a resposta e, a seguir, desvia o programa para a linha 220 a fim de dar continuidade ao laço aberto pela linha 110.

O motivo da inclusão da linha 115 é "zerar" a variável contadora de erros (E) toda vez que uma nova conta é proposta.

A inclusão da linha 235 é necessária para finalizar o programa.

6.1 - O programa pedido é:

```
10 SCREEN 2
20 LINE (10,10)-(131,111),15,BF
30 CIRCLE (71,61),25,8
40 PAINT (71,61),8
50 GOTO 50
```

6.2 - Para desenhar o retângulo pedido rode o seguinte programa:

```
10 SCREEN 2
20 PRESET (100,80)
30 DRAW"C15R30D20L30U20"
```

```
40 GOTO 40
```

A linha 20 serve apenas para posicionar o cursor nas coordenadas da tela em que terá início o desenho feito pelo comando DRAW.

6.3 - a) A parábola fica "mais estreita". Isso ocorre pois a taxa de variação da função aumenta com o aumento do coeficiente do termo de segundo grau.

b) A concavidade da parábola fica voltada "para baixo".

c) As linhas 20 e 30 desenham os eixos. Para desenhá-los em vermelho altere-as para:

```
20 LINE (0,96)-(256,96),8
30 LINE (128,0)-(128,192),8
```

d) Para "congelar" a imagem da tela de alta resolução gráfica.

e) A "densidade" do gráfico aumenta porém, a rapidez na execução do desenho diminui.

7.1 - As alteraçes sugeridas podem ser:

```
15 OPEN "GRP:" AS #1
30 READ A,A$
35 PSET (115,C*10+3),1
36 PRINT #1,A$
80 DATA 6,VERM ESCURO,8,VERMELHO,9,VERM
CLARO,10,AMARELO OURO,11,AMARELO,3,VERDE
 CLARO,2,VERDE,12,VERDE ESCURO,5,AZUL CL
ARO,4,AZUL ESCURO,13,ROXO
```

7.2 - Cancele as linhas 40 e 60 e substitua a de número 50 por:

```
50 LINE(10,C*10+1)-(110,C*10+10),A,BF
```

7.3 a 7.5 - Após fazer esses exercícios você deve obter o programa listado a seguir:

```
100 COLOR 15,1,1
110 SCREEN 2,0
120 X1=64*RND(-TIME)+100:Y1=48*RND(-TIME
)+100:X2=100:Y2=100
130 FOR I=1 TO 8
140 READ A$
150 S$=S$+CHR$(VAL("&B"+A$))
160 NEXT I
170 SPRITE$(1)=S$
180 DATA 00000000
190 DATA 11111111
200 DATA 00000000
210 DATA 11111111
```

```
220 DATA 00011000
230 DATA 11111111
240 DATA 00100100
250 DATA 00000000
260 PUT SPRITE 11,(X1,Y1),8,1
270 FOR I=1 TO 8
280 READ A$
290 W$=W$+CHR$(VAL("&B"+A$))
300 NEXT I
310 SPRITE$(2)=W$
320 DATA 00111000
330 DATA 01010100
340 DATA 10010010
350 DATA 11111110
360 DATA 10010010
370 DATA 01010100
380 DATA 00111000
390 DATA 00000000
400 RESTORE
410 PUT SPRITE 10,(X2,Y2),2,2
420 SPRITE ON
430 IF STRIG(0) THEN N=N-1:PLAY"S8M8001C
4"
440 X1=X1+3*INT(RND(1)+.5)-3*INT(RND(1)+
.5)
450 Y1=Y1+3*INT(RND(1)+.5)-3*INT(RND(1)+
.5)
460 IF X1<0 OR X1>255 OR Y1<0 OR Y1>192
THEN X1=128:Y1=96
470 A=STICK(0)
480 Y2=Y2+(A=8)+(A=1)+(A=2)-(A=6)-(A=5)-
(A=4)
490 X2=X2+(A=8)+(A=7)+(A=6)-(A=2)-(A=3)-
(A=4)
500 PUT SPRITE 10,(X2,Y2),2,2
510 PUT SPRITE 11,(X1,Y1),8,1
520 ON SPRITE GOSUB 570
530 B=B+1
540 IF B>200 THEN 600
550 GOTO 420
```

```
560 END
570 IF STRIG(0) THEN PLAY"","V15O5C16":N
=N+4:PSET(X1+4,Y1+4)
580 SPRITE OFF
590 RETURN
600 OPEN"GRP:"AS#1
610 PSET(10,10),1
620 PRINT#1,"VOCE FEZ";N;"PONTOS!"
630 FORI=1 TO 3000:NEXT I
640 RUN
```

7.6 - a) Apertando o botão 1 do joystick A.
b) Alterar para N=N-3.

7.7 - a) Trocar todos os algarismos 3 por 6.
b) Trocar todos os algarismos 3 (ou 6, se você já fez a alteração do item anterior) por 1.

7.8 - Fazer as seguintes alterações:

```
120 X1=64*RND(-TIME):Y1=48*RND(-TIME):X2
=32:Y2=24
460 IF X1<0 OR X1>64 OR Y1<0 OR Y1>48 TH
EN X1=32:Y1=24
```

7.9 - Alterar a linha 470 para:

```
470 A=STICK(1)
```

7.10- Alterar para:

```
540 IF B>100 THEN 600
```

7.11- a) Trocar STRIG(0) por STRIG(1)
b) Alterar, na linha 570, N=N+4 para N=N+10.

7.12- a) Mude a linha 610 para:

```
610 PSET(50,96),1
```

b) Se a alteração do item anterior não tiver sido efetuada a linha 625 deverá ser:

```
625 IF N>100 THEN PSET(10,20),1:PRINT #1
,"PARABENS!"
```

c) Alterar para:

```
630 FOR I=1 TO 1500:NEXT I
```

7.13- a) Os SPRITES aparecem ampliados.
b) Alterar a linha 570 para:

```
570 IF STRIG(0) THEN PLAY"","V15o5C16":N
=N+100:PSET(X2+8,Y2+8)
```

8.1 - As linhas 10 e 20 são:

```
10 PLAY"S0M5000o4"
20 PLAY"E16F16G16G8G16F8E8A8A4"
```

8.2 - A linha 30 é:

```
30 PLAY"D16E16F16F8F16E8D8G4."
```

8.3 - A linha 40 é:

```
40 PLAY"E8G16G8G16F8E8A8o5C8"
```

8.4 - A linha 50 é:

```
50 PLAY"o4B8A8G8E8F8D8C4."
```

8.5 - A primeira versão consome menos memória.

8.6 - A música completamente transcrita fica:

```
5 PLAY "V15T180","V12T180","V10T180"
10 PLAY "o4A4","o4R4","o4R4"
20 PLAY "o5C2D4","A2.","E2."
30 PLAY "E4.F8E4","o5C2.","A2."
40 PLAY "D2o4B4","o4G2.","B2G4"
50 PLAY "G4.A8B4","B4.A8G4","D2."
60 PLAY "o5C2o4A4","A2.","E2."
70 PLAY "A4.G+8A4","F2.","D2A4"
80 PLAY"B2G+4","E4o5E4o4B4","G+2E4"
90 PLAY"E2A4","E2R8","G+2R8"
100 PLAY"o5C2D4","A2.","E2."
110 PLAY"E4.F8E8","o5C2.","A2."
120 PLAY"D2o4B4","o4G2.","B2G4"
130 PLAY"G4.A8B4","B4.A8G4","D2."
140 PLAY"o5C4.o4B8A4","A4.B8o5C8D8","E2.
"
150 PLAY"G+4.F+8G+4","E2o4E4","D2."
160 PLAY"A2.","A4o5A4E4","C2."
170 PLAY"o4A2.","o4A2.","C2."
```

8.7 - A transcrição é:

A "Série Didática" é mais uma iniciativa pioneira da EDITORA ALEPH visando levar aos seus leitores conceitos específicos através de textos claros e, realmente, didáticos. O uso do microcomputador na educação pode ser desastroso ou extremamente proveitoso, dependendo da forma como ele for usado. Na "Série Didática" a utilização dos micros é o tema central, sendo repleta de exemplos de usos proveitosos.

curso de basic

Este livro destina-se àqueles que querem se iniciar em programação utilizando o BASIC MSX. Escrito por professores de larga experiência, foi estruturado em verdadeiras aulas cujo nível de dificuldade cresce com o conhecimento do leitor.

A teoria é apresentada de maneira clara e didática, sempre enriquecida por exemplos que devem ser testados no computador para que o leitor possa, através de uma interação com a máquina, tirar o máximo proveito do que está sendo abordado.

ALEPH PUBLICAÇÕES E ASSESSORIA PEDAGÓGICA LTDA.
CP. 20.707 - CEP. 01498 - SÃO PAULO

A "Série Didática" é mais uma iniciativa pioneira da EDITORA ALEPH visando levar aos seus leitores conceitos específicos através de textos claros e, realmente, didáticos. O uso do microcomputador na educação pode ser desastroso ou extremamente proveitoso, dependendo da forma como ele for usado. Na "Série Didática" a utilização dos micros é o tema central, sendo repleta de exemplos de usos proveitosos.

curso de basic

Este livro destina-se àqueles que querem se iniciar em programação utilizando o BASIC MSX. Escrito por professores de larga experiência, foi estruturado em verdadeiras aulas cujo nível de dificuldade cresce com o conhecimento do leitor.

A teoria é apresentada de maneira clara e didática, sempre enriquecida por exemplos que devem ser testados no computador para que o leitor possa, através de uma interação com a máquina, tirar o máximo proveito do que está sendo abordado.

ALEPH PUBLICAÇÕES E ASSESSORIA PEDAGÓGICA LTDA.
CP. 20.707 - CEP. 01498 - SÃO PAULO